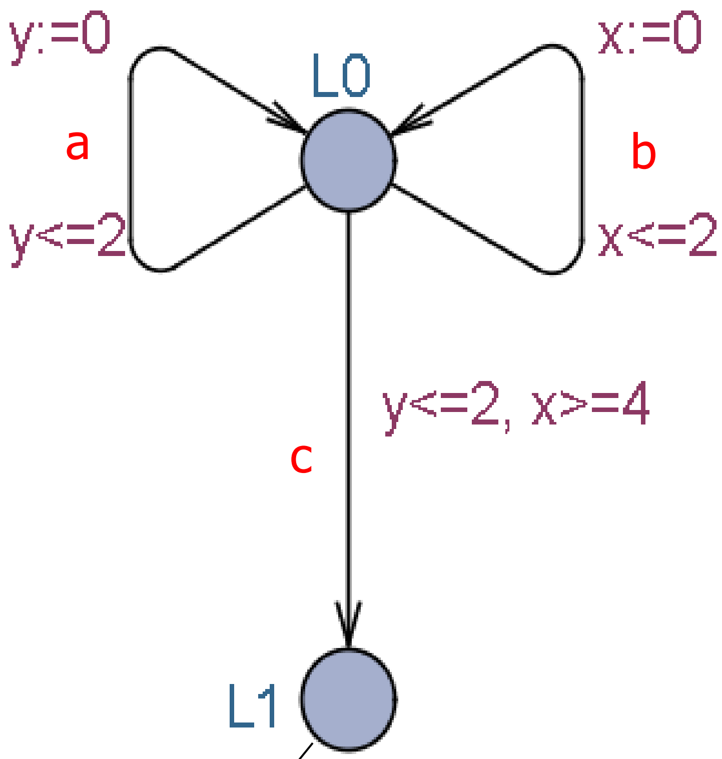
Decidability and Symbolic Verification

Kim G. Larsen
Aalborg University, DENMARK

Decidability

Reachability ?



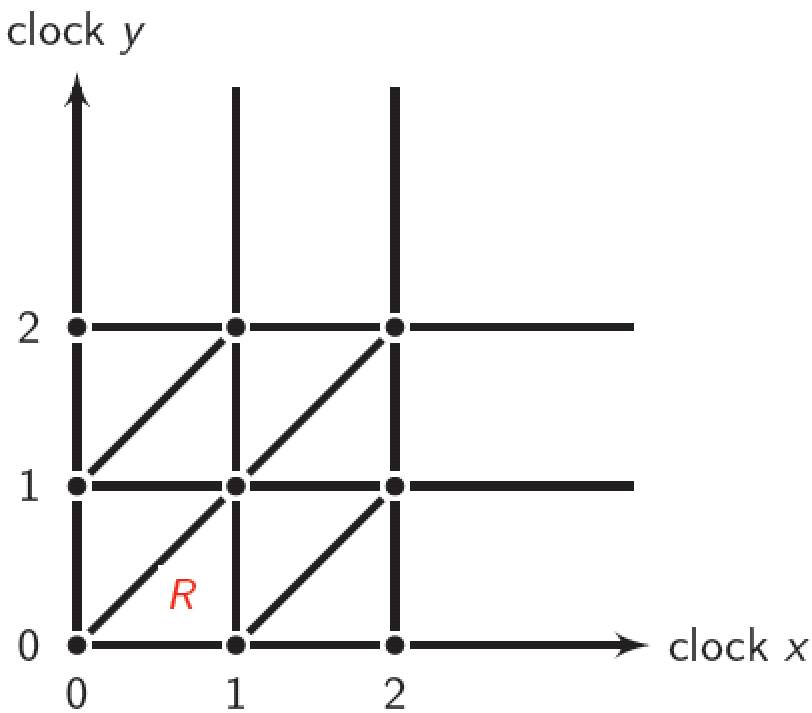
OBSTACLE:
Uncountably infinite
state space

$L \times \mathbb{R}^C$

locations — clock-valuations

Reachable from initial state (L0,x=0,y=0) ?

The Region Abstraction



region R defined by:

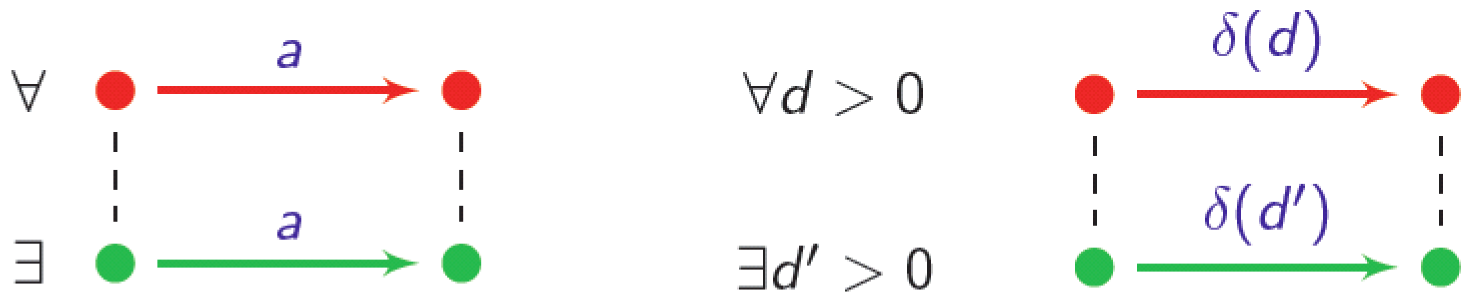
$$\begin{cases} 0 < x < 1 \\ 0 < y < 1 \\ y < x \end{cases}$$

- "compatibility" between regions and constraints
- "compatibility" between regions and time elapsing

$\rightsquigarrow$ an equivalence of finite index
a time-abstract bisimulation

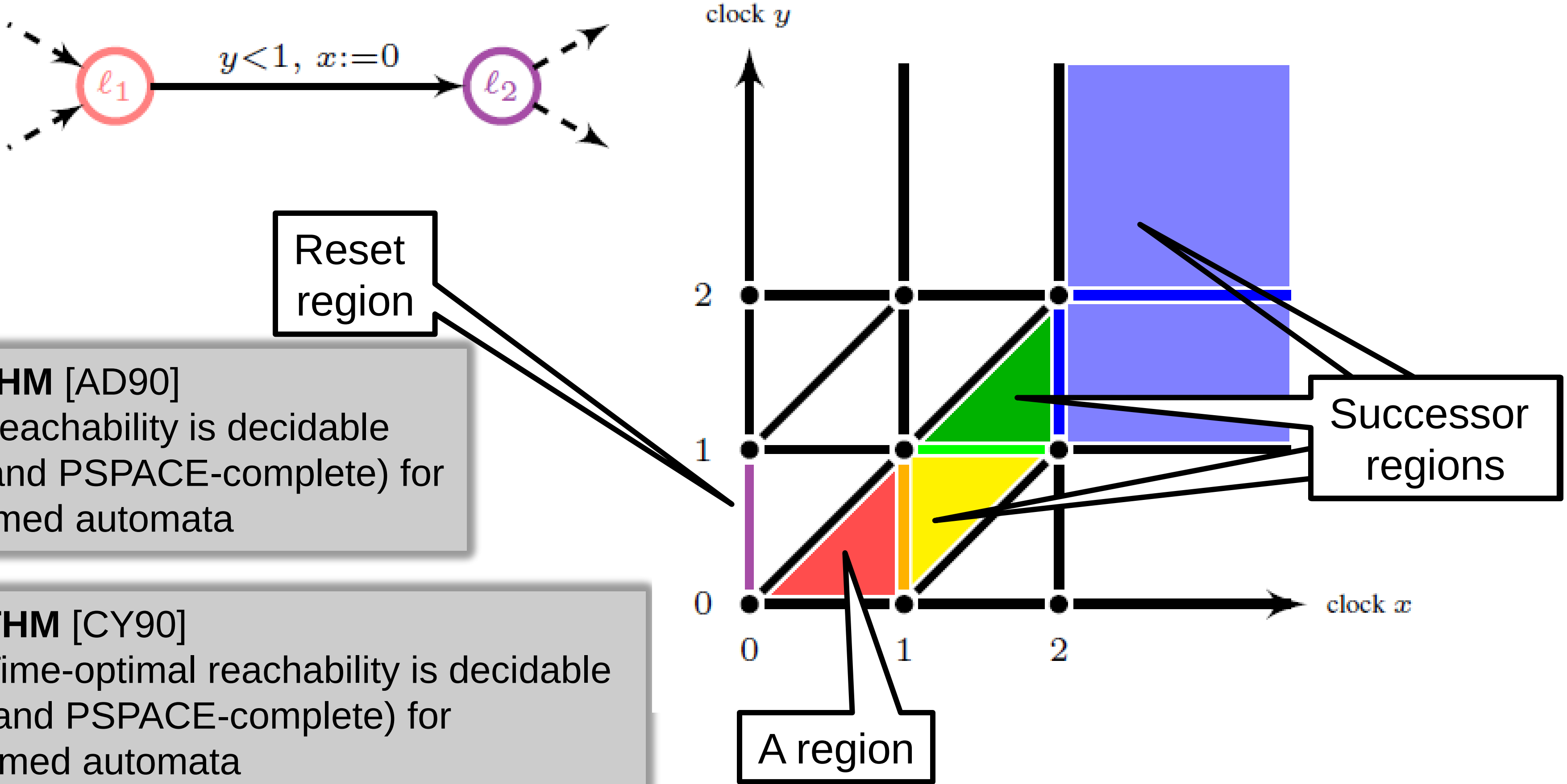
Time Abstracted Bisimulation

This is a relation between • and • such that:



... and vice-versa (swap • and •).

Regions – From Infinite to Finite

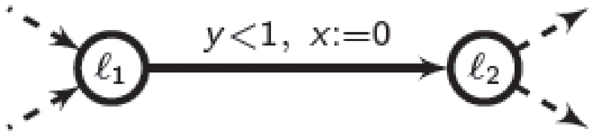


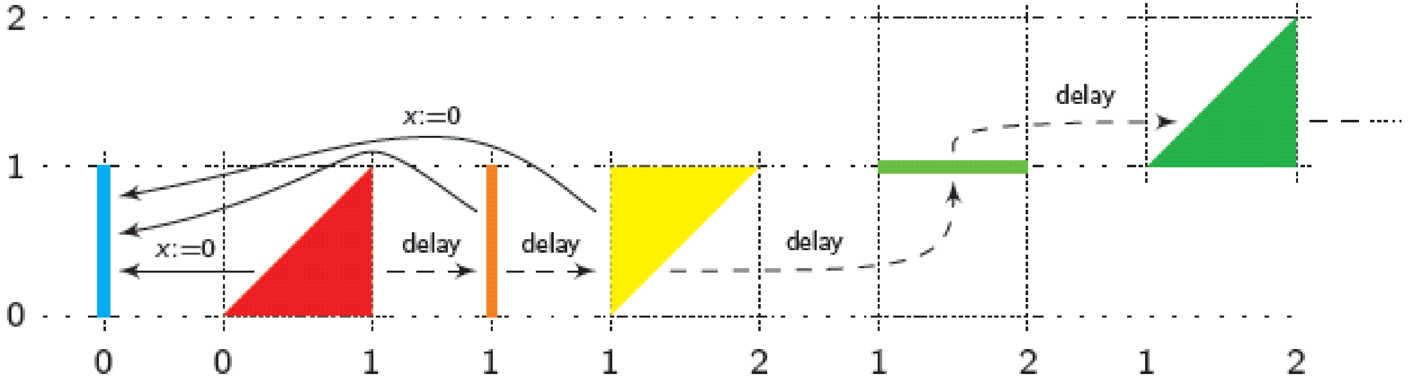
THM [AD90]
Reachability is decidable (and PSPACE-complete) for timed automata

THM [CY90]
Time-optimal reachability is decidable (and PSPACE-complete) for timed automata

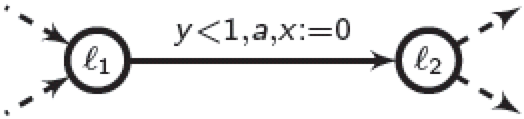
Region Graph

It "mimicks" the behaviours of the clocks.



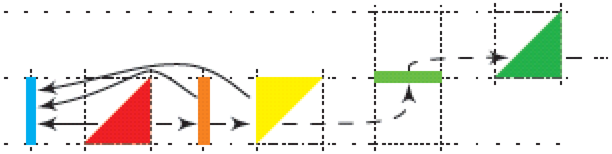


Region Automaton = Finite Bisimulation Quotiont



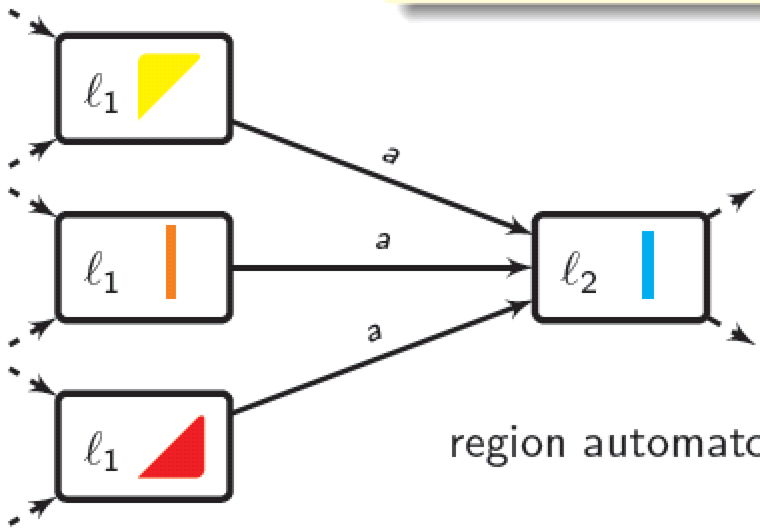
timed automaton

$\otimes$



region graph

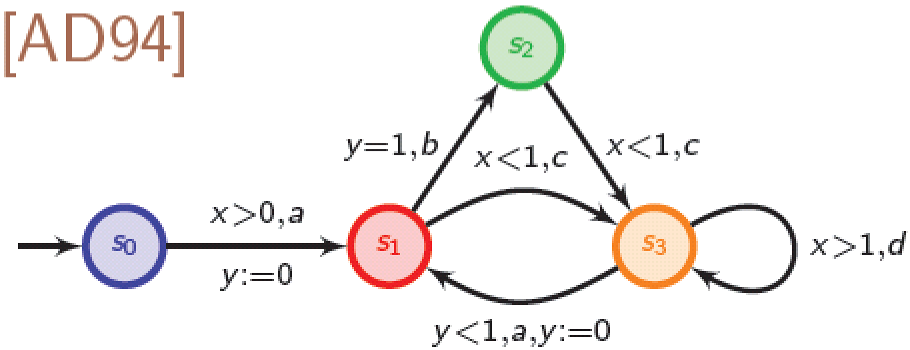
$\mathcal{L}$(reg. aut.) = UNTIME($\mathcal{L}$(timed aut.))

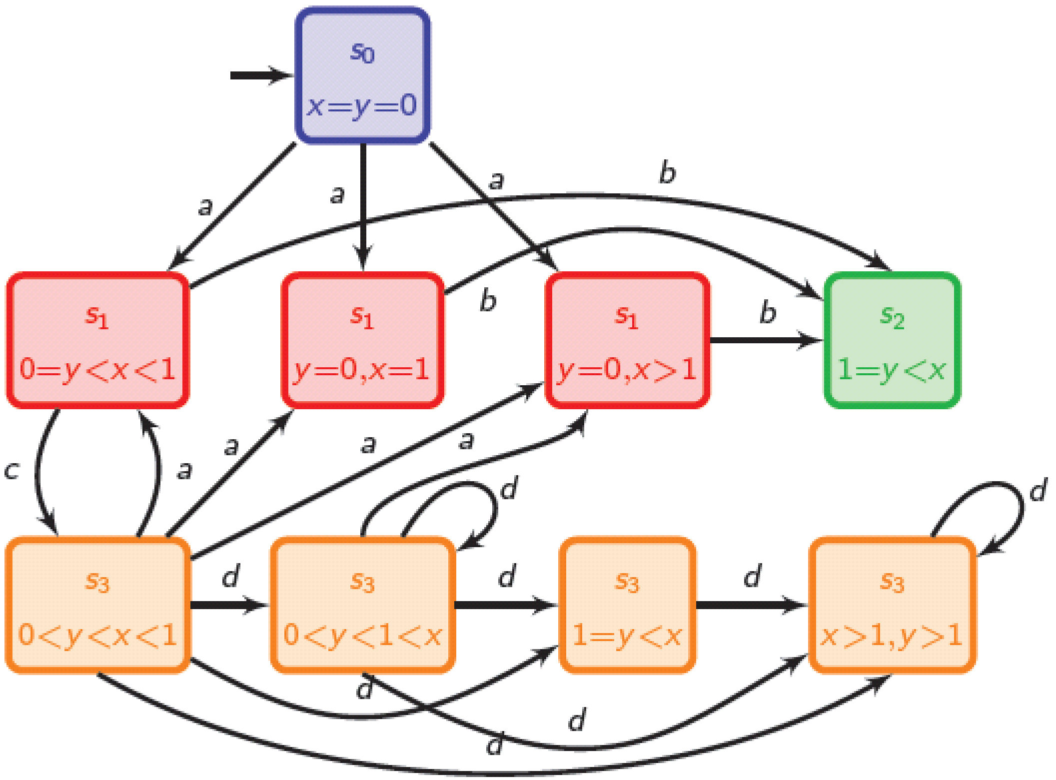


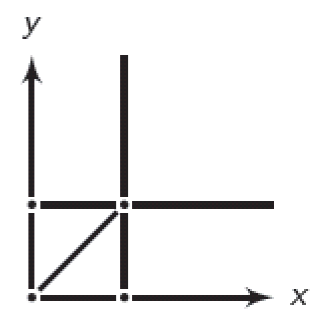
region automaton

An Example

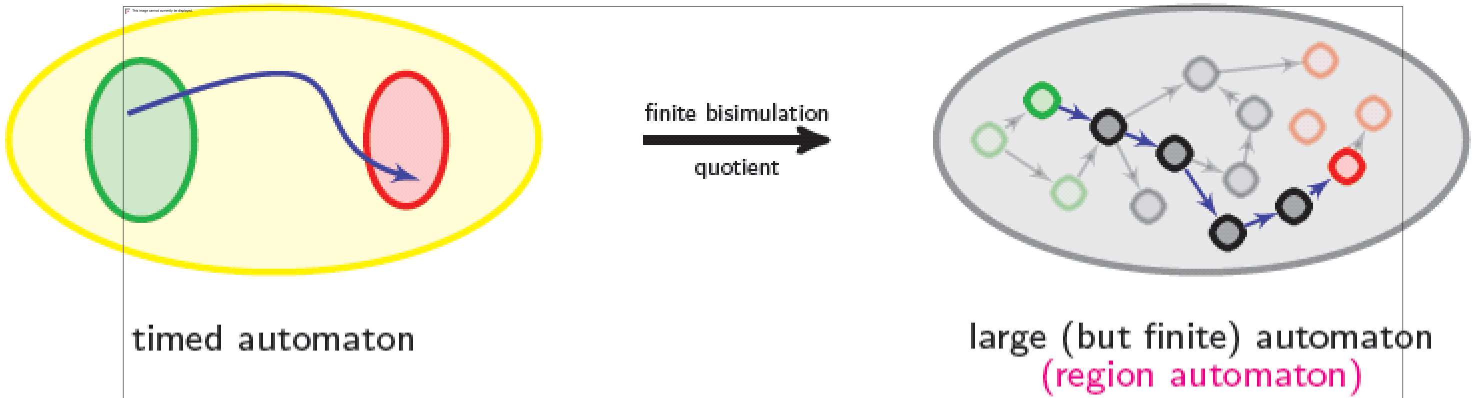
[AD94]







Region Automaton



LARGE: exponential in the number of clocks and in the constants (if encoded in binary). The number of regions is

$$\prod_{x \in X} (2M_x + 2) \cdot |X!| \cdot 2^{|X|}$$

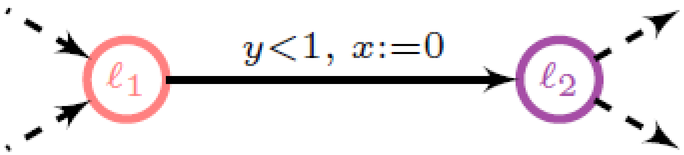
Fundamental Results

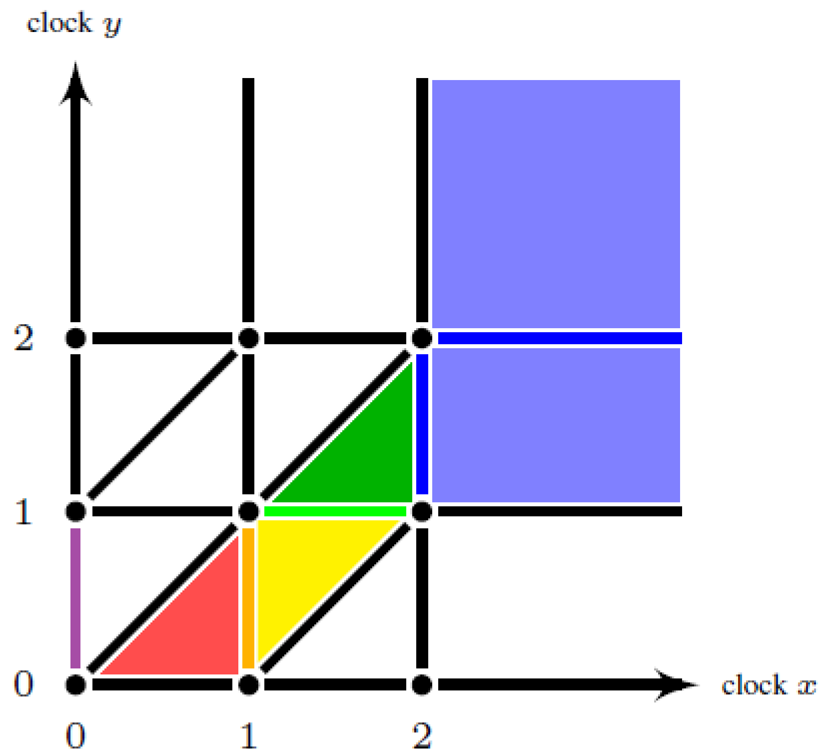
- Reachability ☺ PSPACE-c
- Model-checking
  - TCTL ☺ PSPACE-c ; MTL ☹ UNDECIDABLE ; MITL ☺ EXPSPACE-c
- Bisimulation, Simulation
  - Timed ☺ EXPTIME-c ; Untimed ☺
- Trace-inclusion
  - Timed ☹ UNDECIDABLE ; Untimed ☺ EXPSPACE-c

Symbolic Verification

The UPPAAL Verification Engine

Regions – From Infinite to Finite



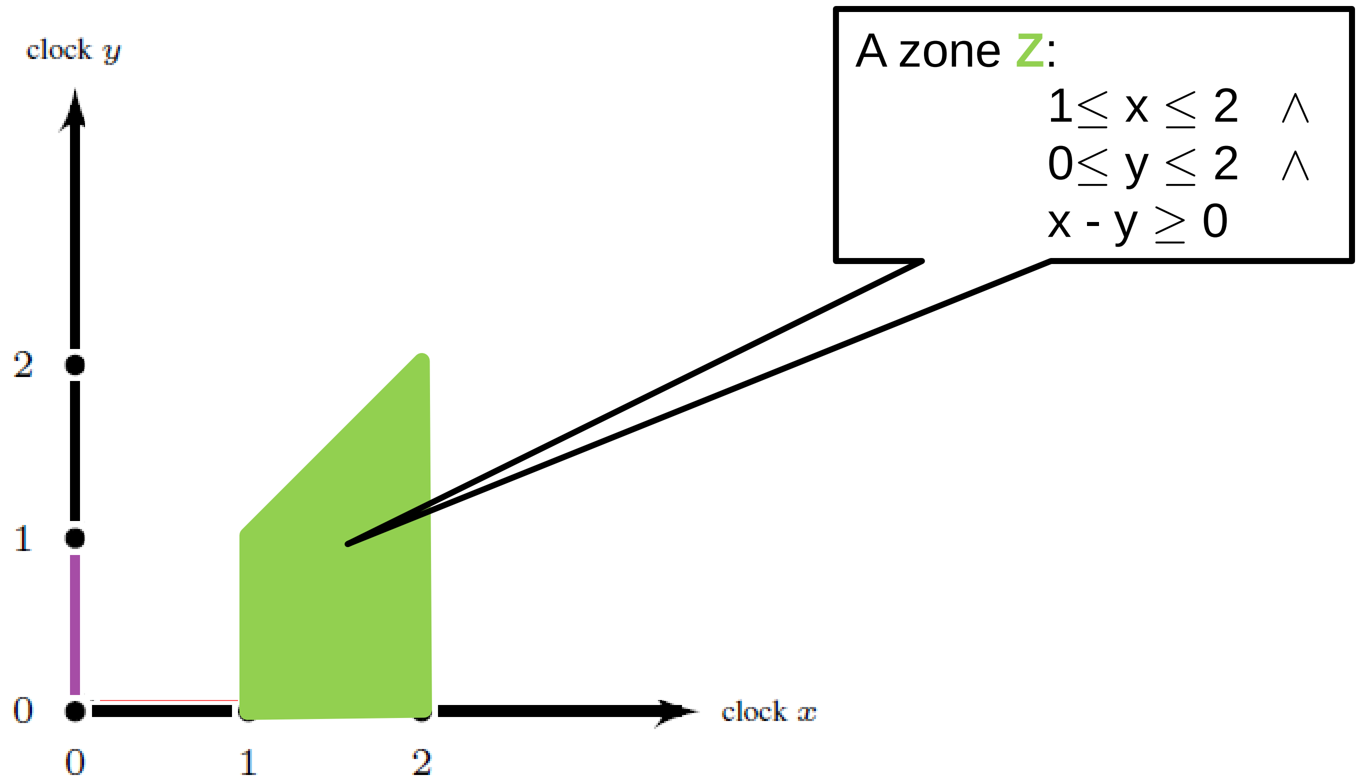


Theorem

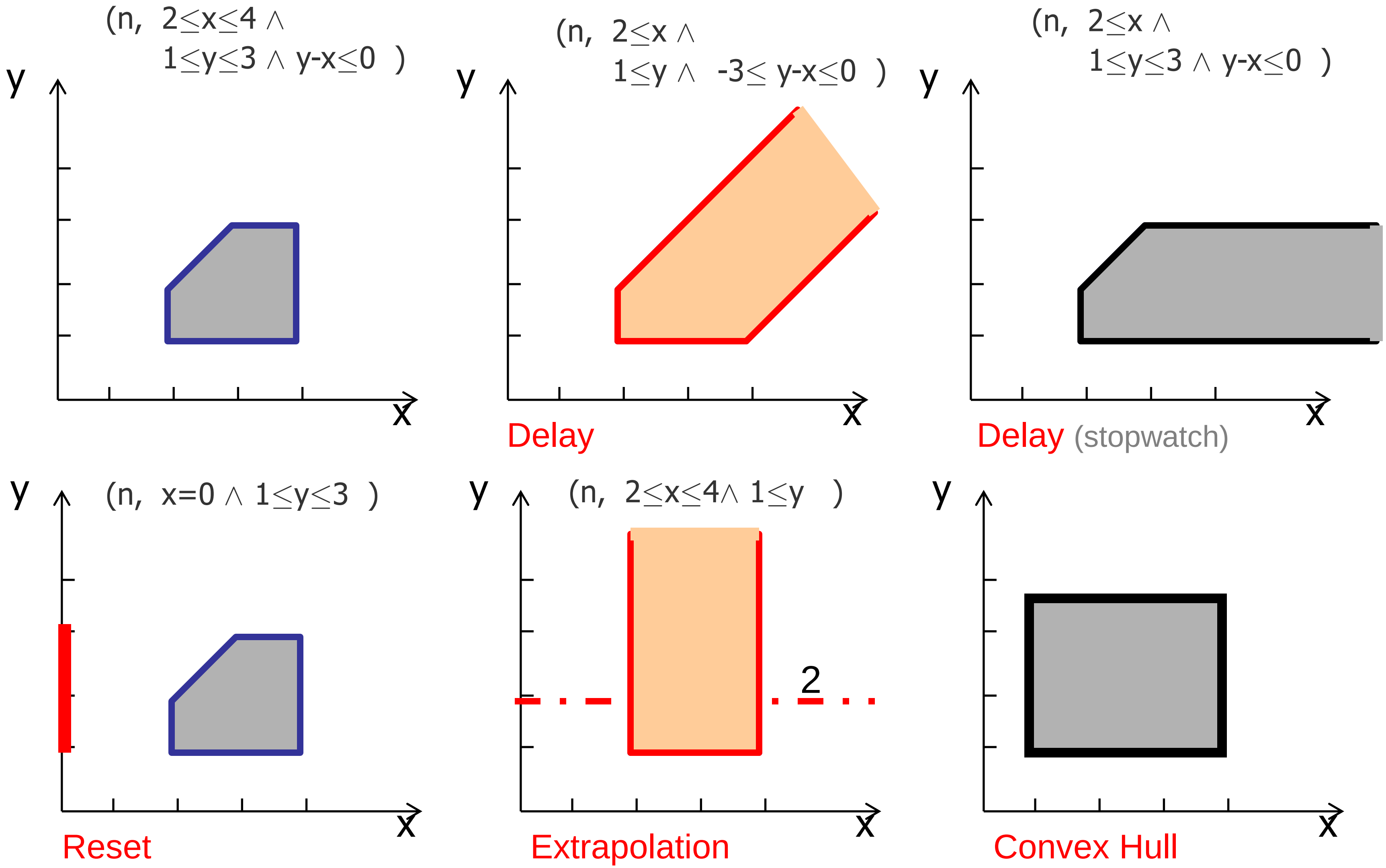
The number of regions is $n! \cdot 2^n \cdot \prod_{x \in C}(2c_x + 2)$.

Region construction: [AD94]
In practice: Zones

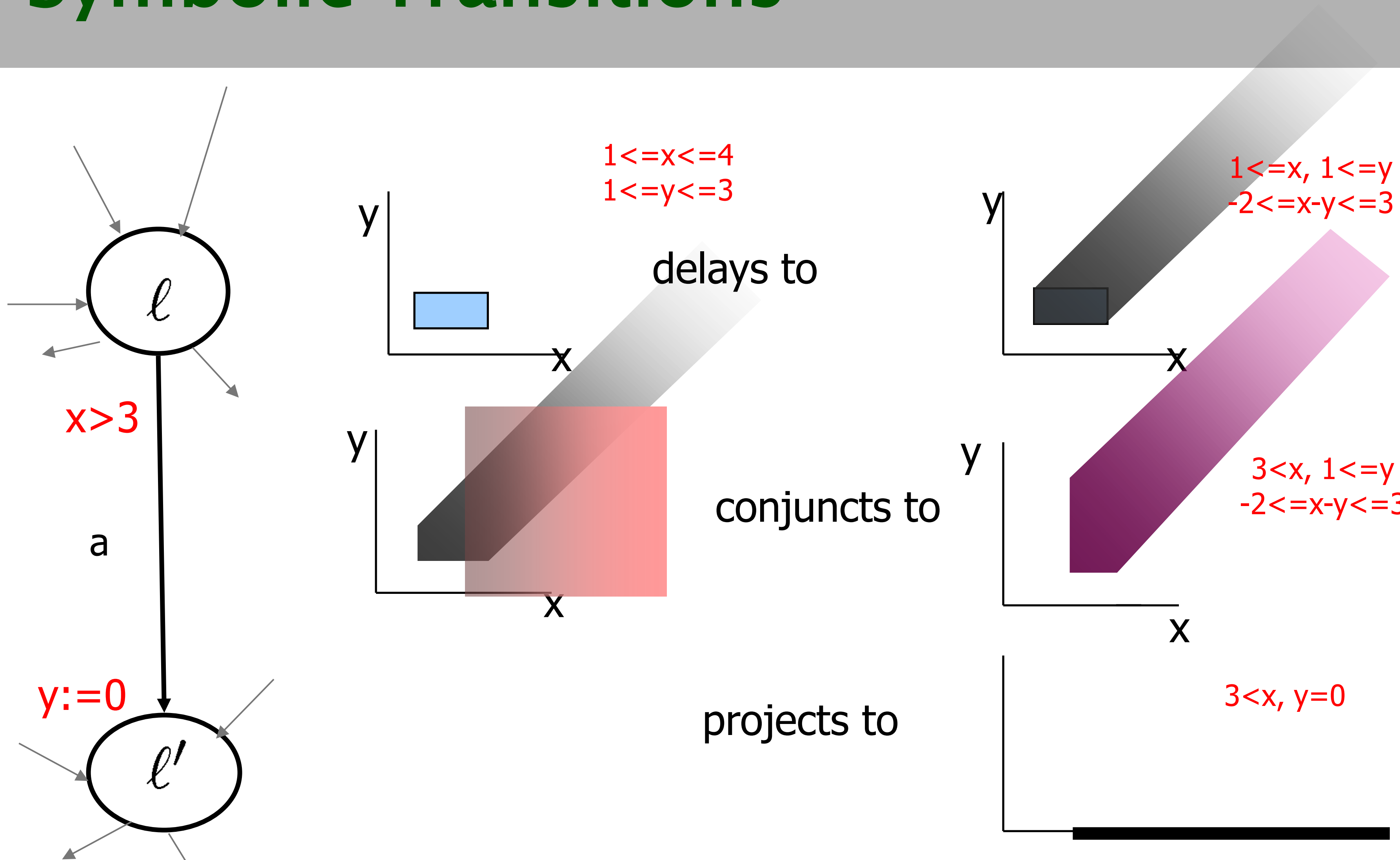
Zones – From Finite to Efficiency



Zones – Operations



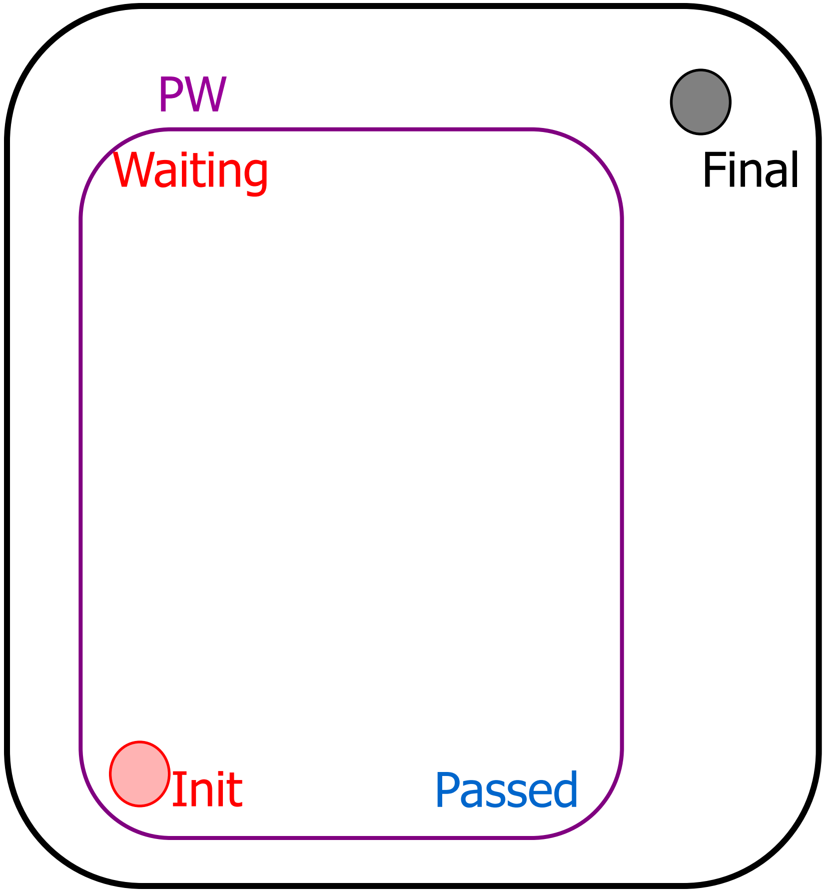
Symbolic Transitions



$$(\ell, 1 \leq x \leq 4, 1 \leq y \leq 3) \Rightarrow_a (\ell', 3 < x, y = 0)$$

Forward Reachability

Init -> Final ?



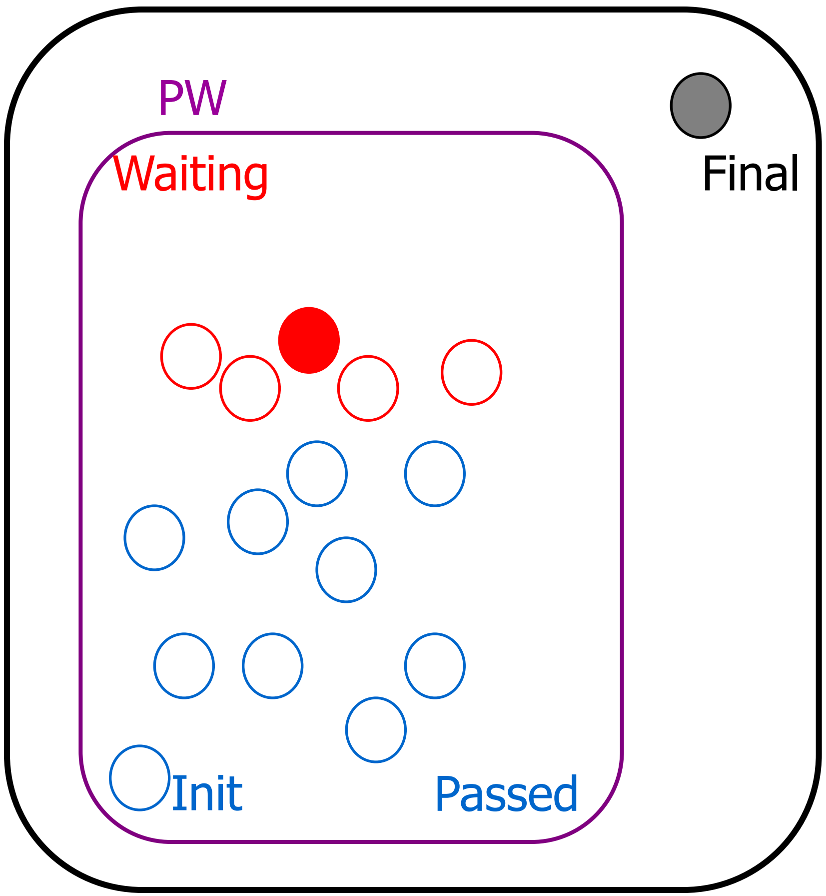
INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
- pick (n,Z) in Waiting
- if (n,Z) = Final return true
- for all (n,Z)→(n',Z'):
  - if for some (n',Z'') Z' ⊆ Z'' continue
  - else add (n',Z') to Waiting
  - move (n,Z) to Passed

UNTIL Waiting = Ø
return false

Forward Reachability

Init -> Final ?



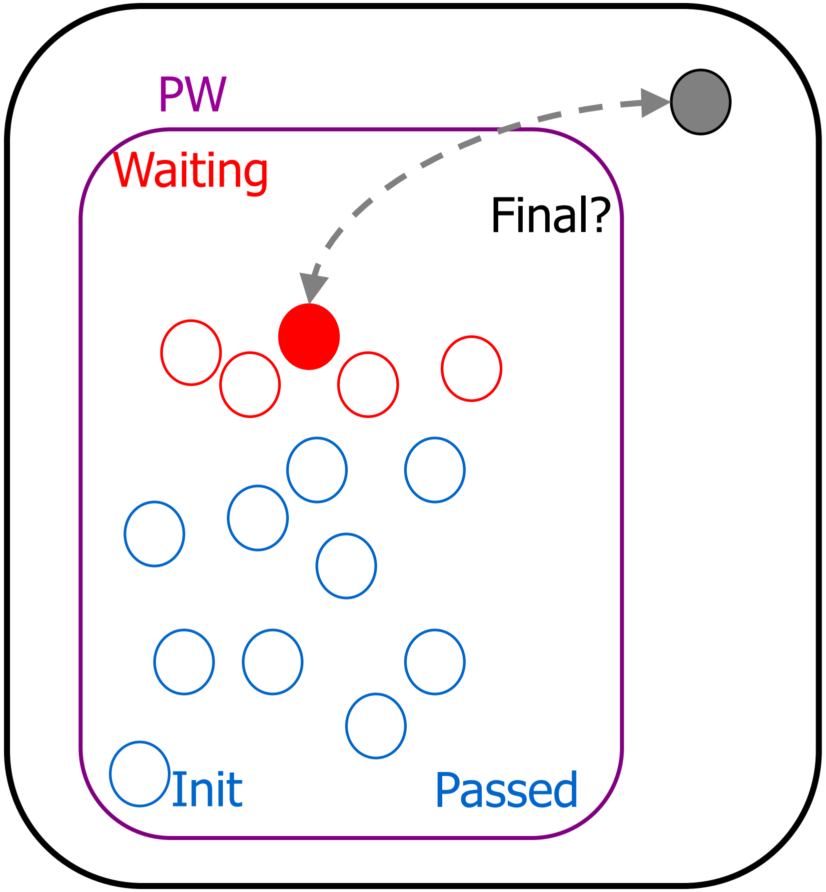
INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z)→(n',Z'):
if for some (n',Z'') Z'⊆ Z'' continue
else add (n',Z') to Waiting
move (n,Z) to Passed

UNTIL Waiting = Ø
return false

Forward Reachability

Init -> Final ?



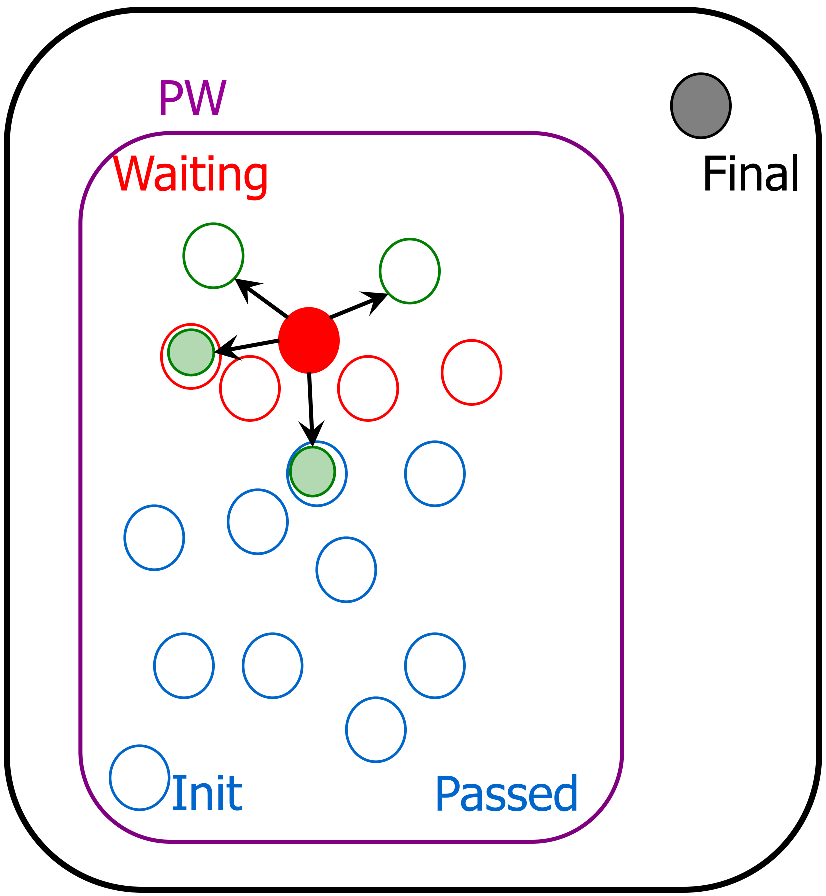
INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z)→(n',Z'):
if for some (n',Z'') Z'⊆ Z'' continue
else add (n',Z') to Waiting
move (n,Z) to Passed

UNTIL Waiting = Ø
return false

Forward Reachability

Init -> Final ?



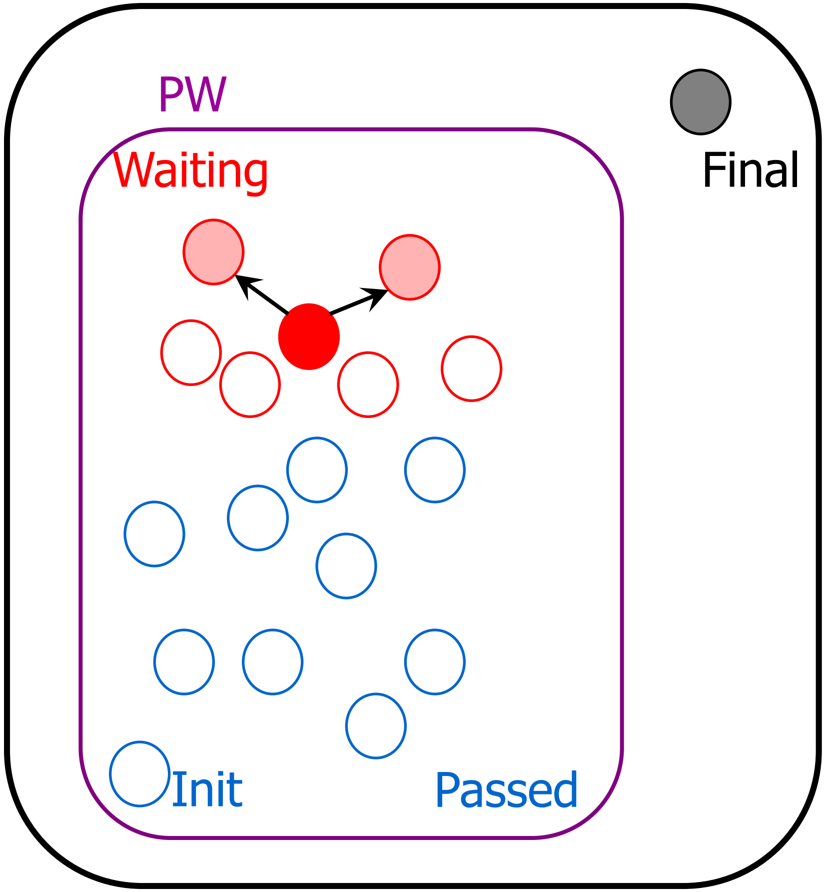
INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z)→(n',Z'):
if for some (n',Z'') Z'⊆ Z'' continue
else add (n',Z') to Waiting
move (n,Z) to Passed

UNTIL Waiting = Ø
return false

Forward Reachability

Init -> Final ?



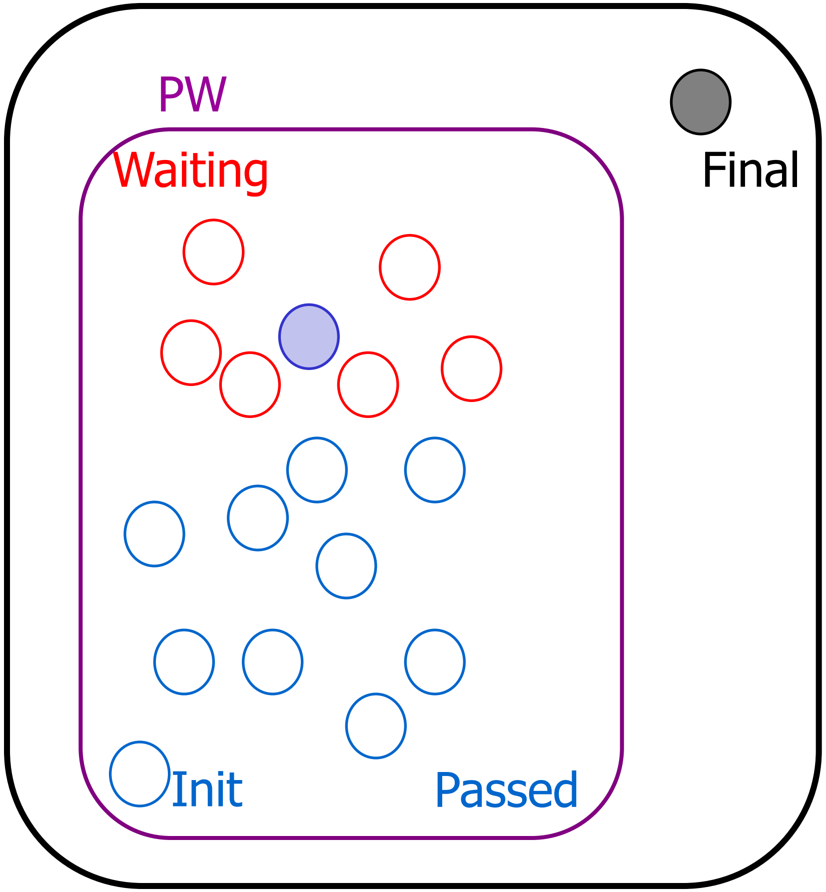
INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
- pick (n,Z) in Waiting
- if (n,Z) = Final return true
- for all (n,Z)→(n',Z'):
  - if for some (n',Z'') Z'⊆ Z'' continue
  - else add (n',Z') to Waiting
  - move (n,Z) to Passed

UNTIL Waiting = Ø
return false

Forward Reachability

Init -> Final ?



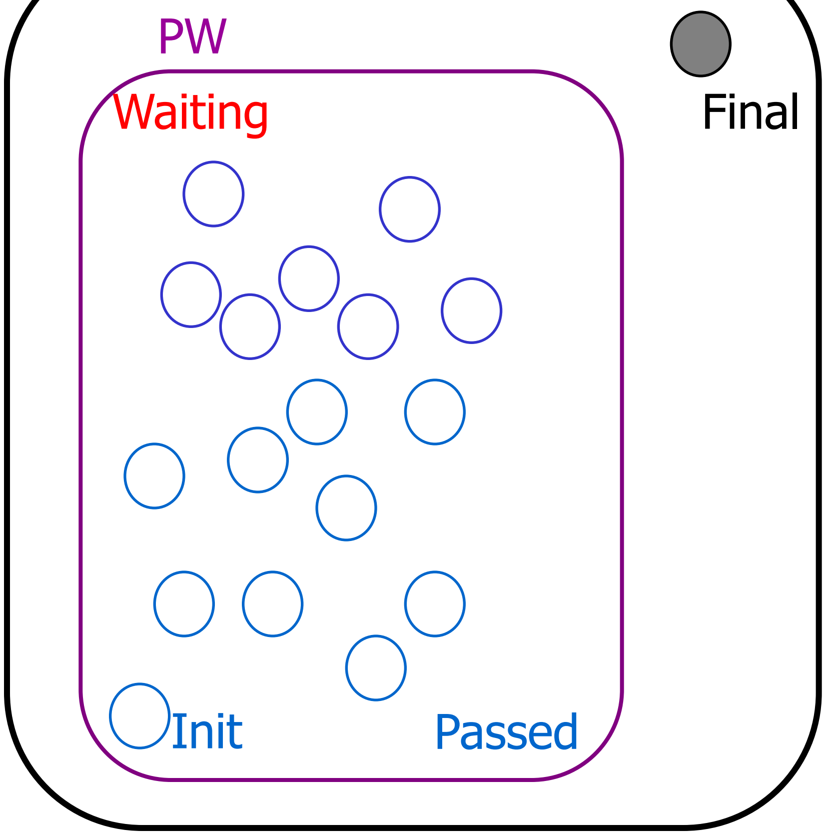
INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z)→(n',Z'):
if for some (n',Z'') $Z' \subseteq Z''$ continue
else add (n',Z') to Waiting
move (n,Z) to Passed

UNTIL Waiting = Ø
return false

Forward Reachability

Init -> Final ?

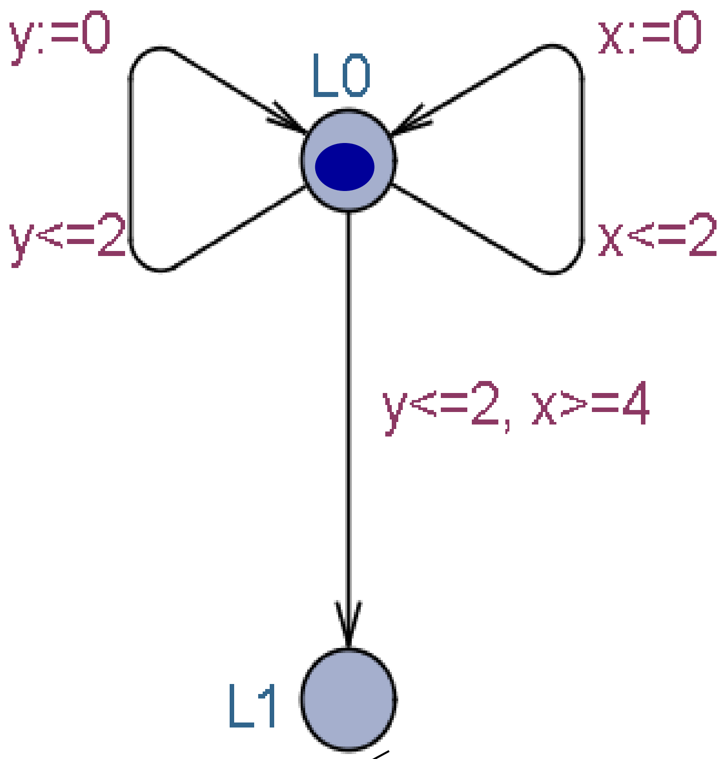


INITIAL Passed := Ø;
 Waiting := {(n_0,Z_0)}

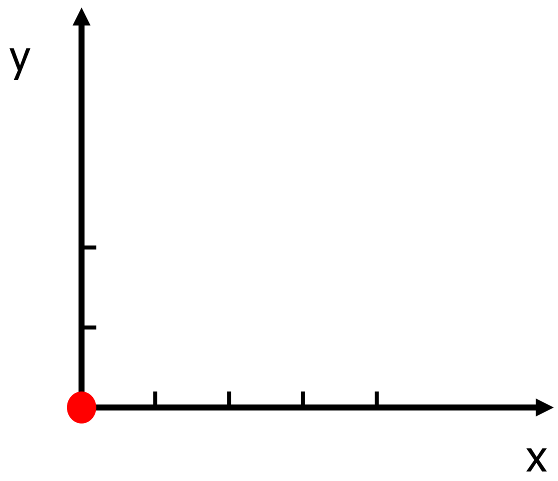
REPEAT
 pick (n,Z) in Waiting
 if (n,Z) = Final return true
 for all (n,Z)→(n',Z'):
  if for some (n',Z'') Z' ⊆ Z'' continue
  else add (n',Z') to Waiting
  move (n,Z) to Passed

UNTIL Waiting = Ø
return false

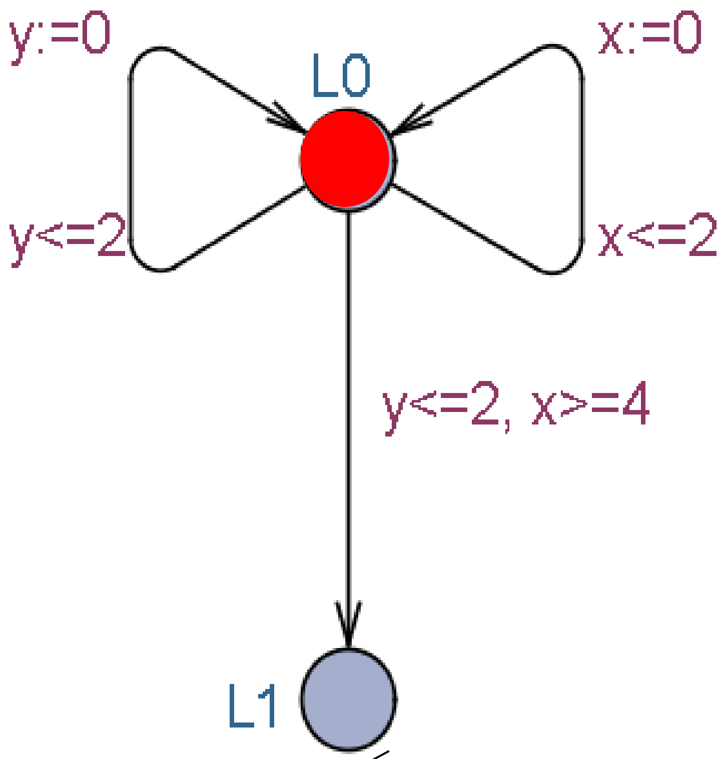
Symbolic Exploration



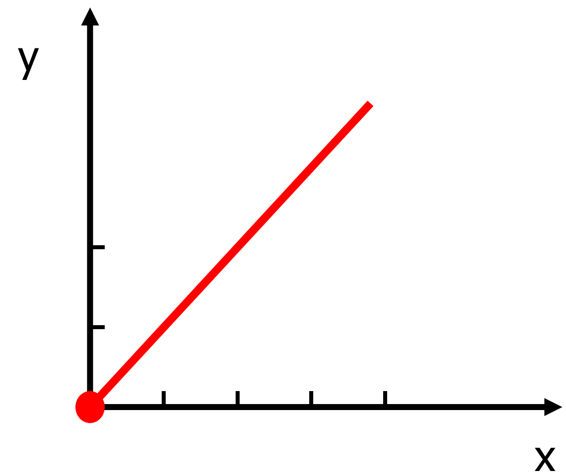
Reachable?



Symbolic Exploration

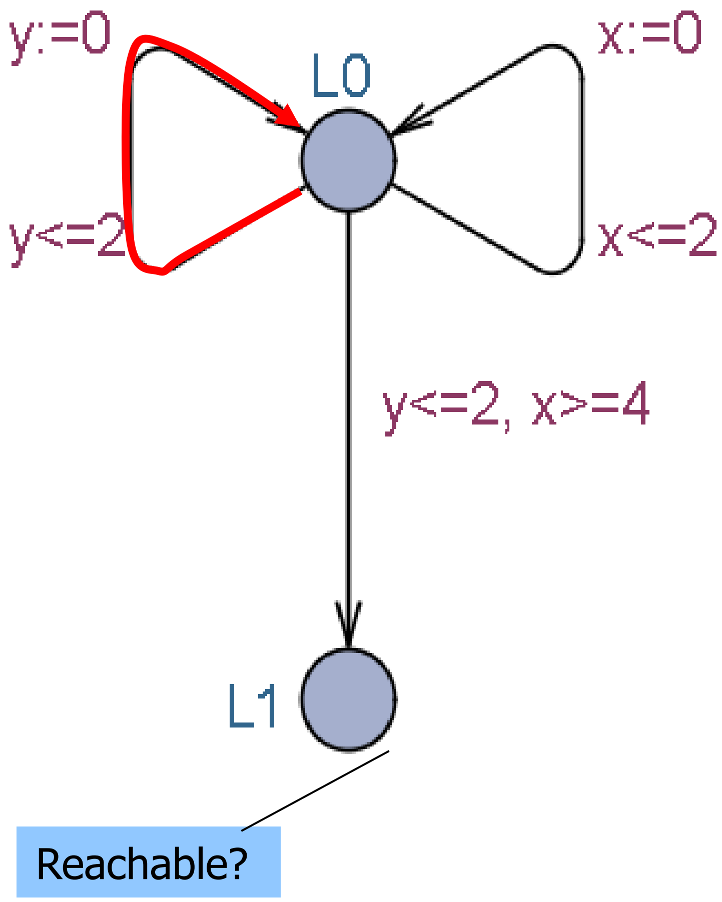


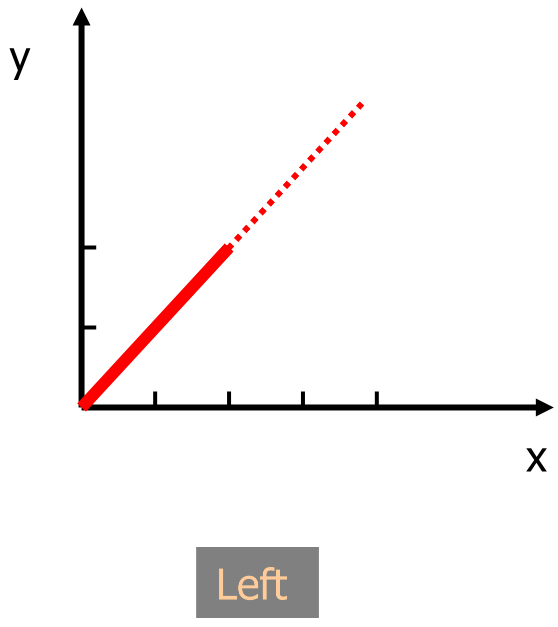
Reachable?



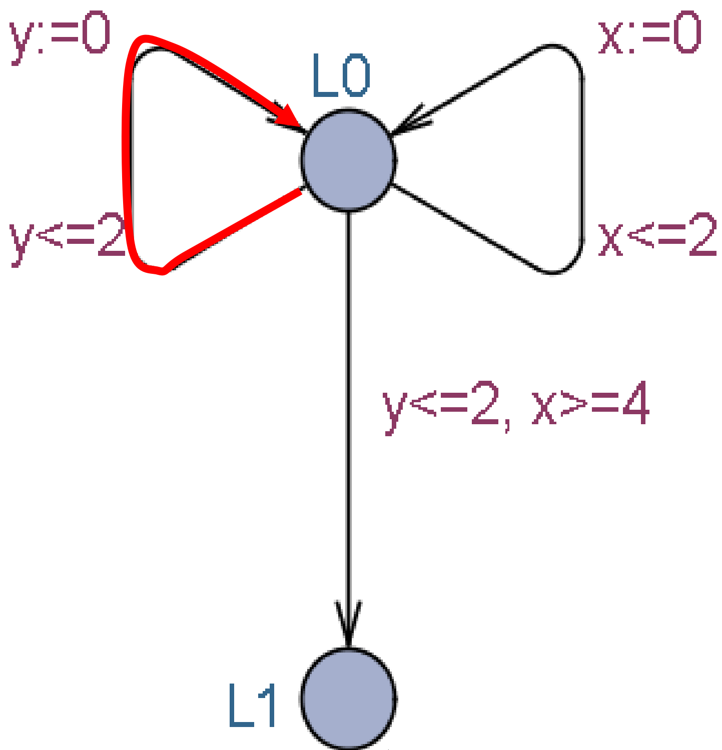
Delay

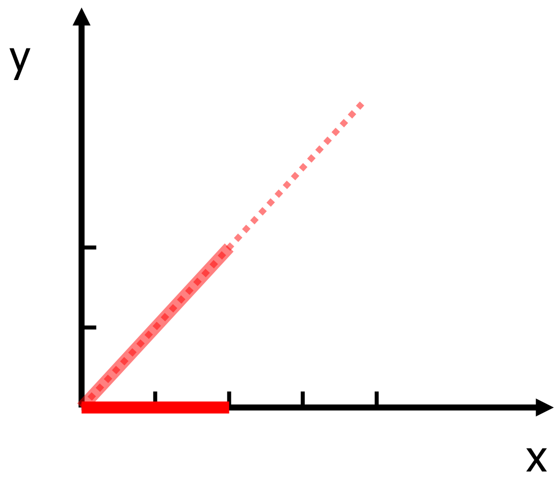
Symbolic Exploration





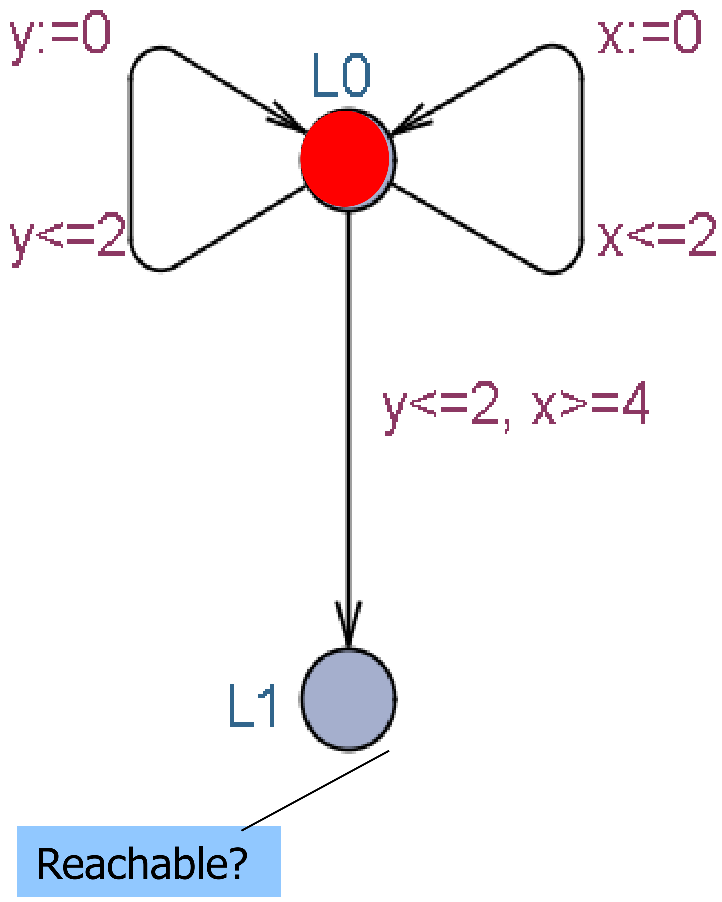
Symbolic Exploration

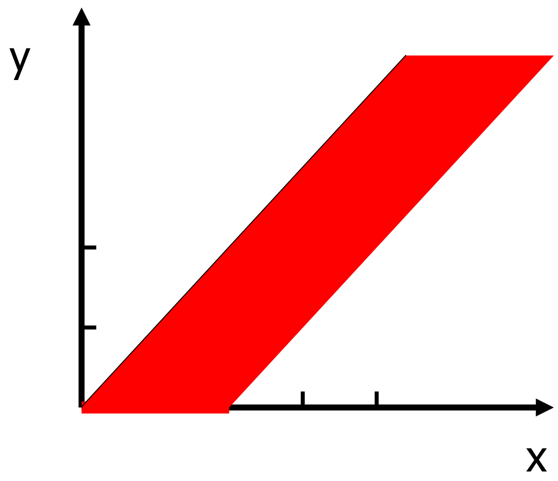




Left

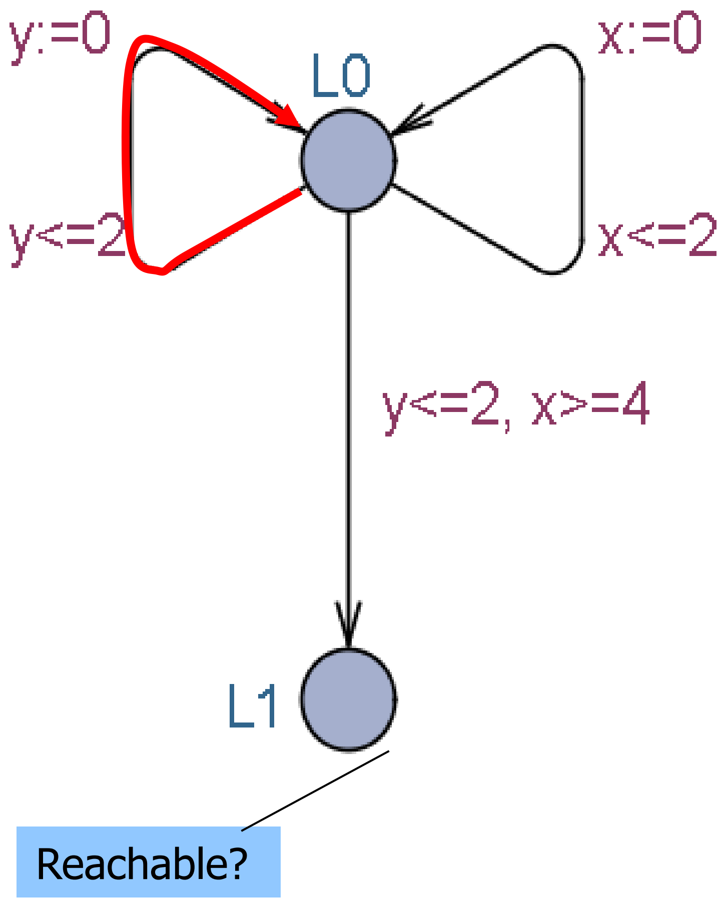
Symbolic Exploration

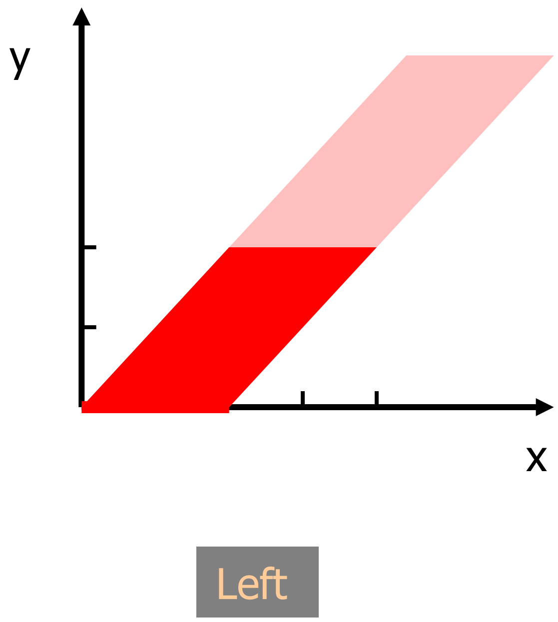




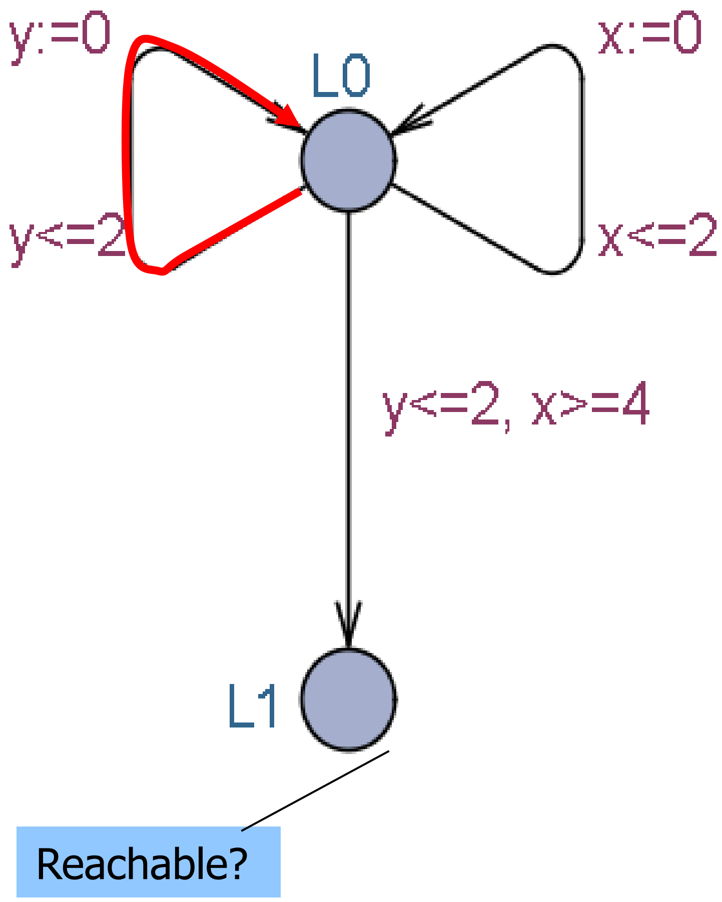
Delay

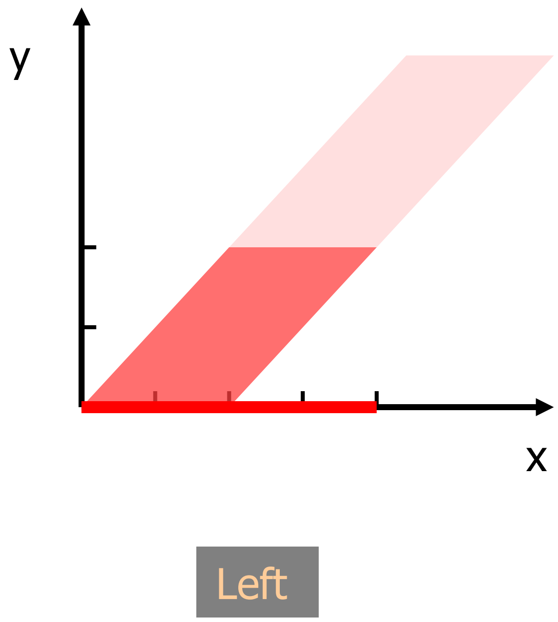
Symbolic Exploration



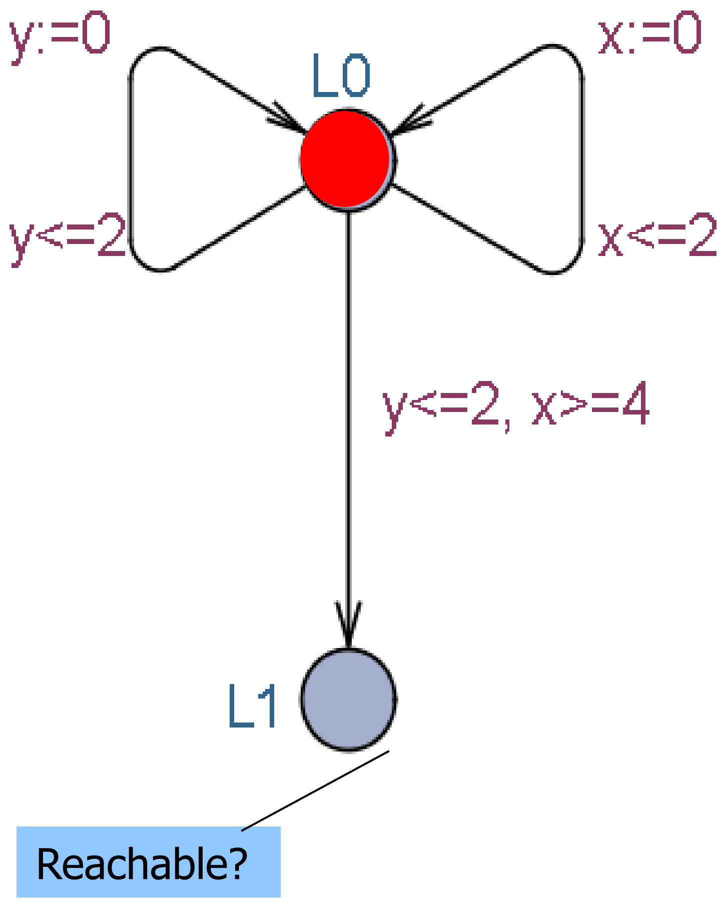


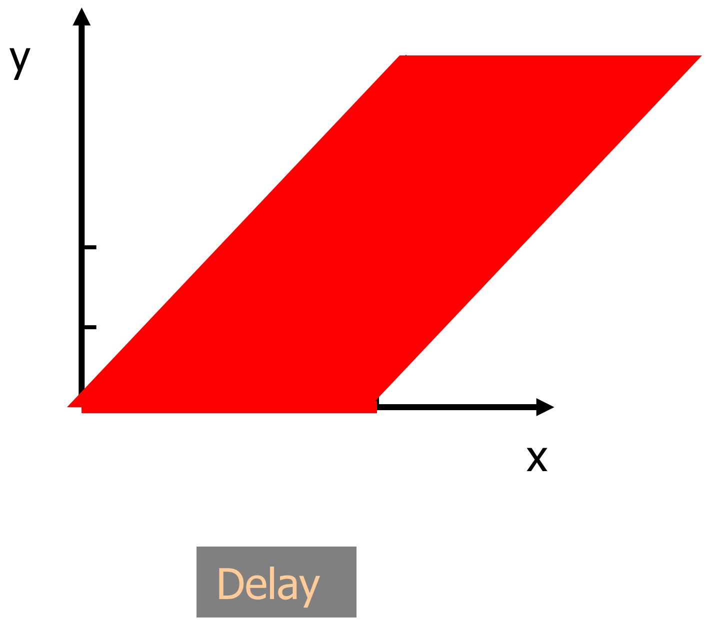
Symbolic Exploration



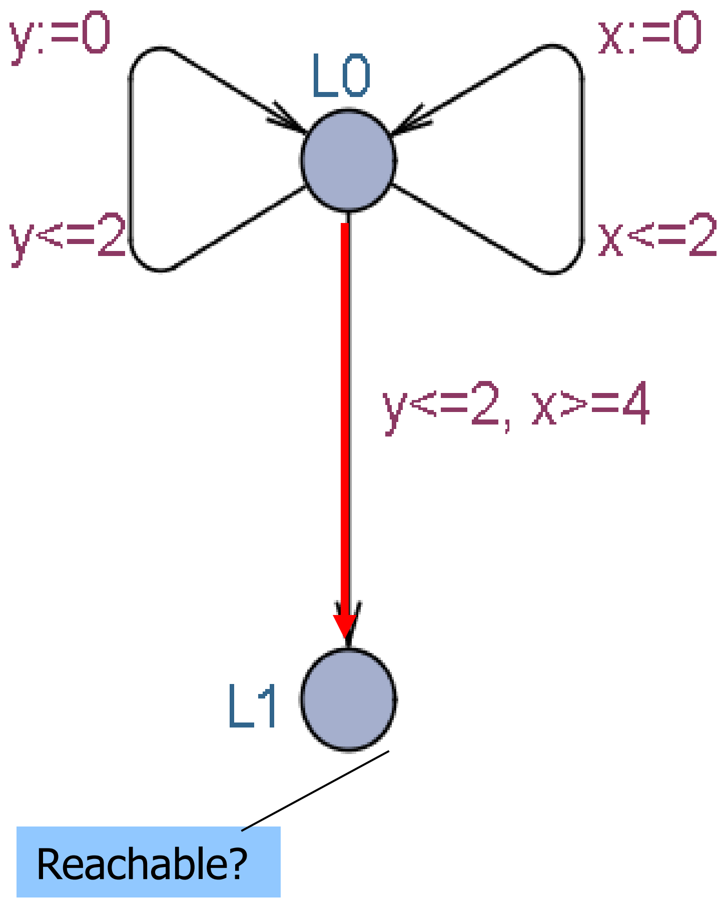


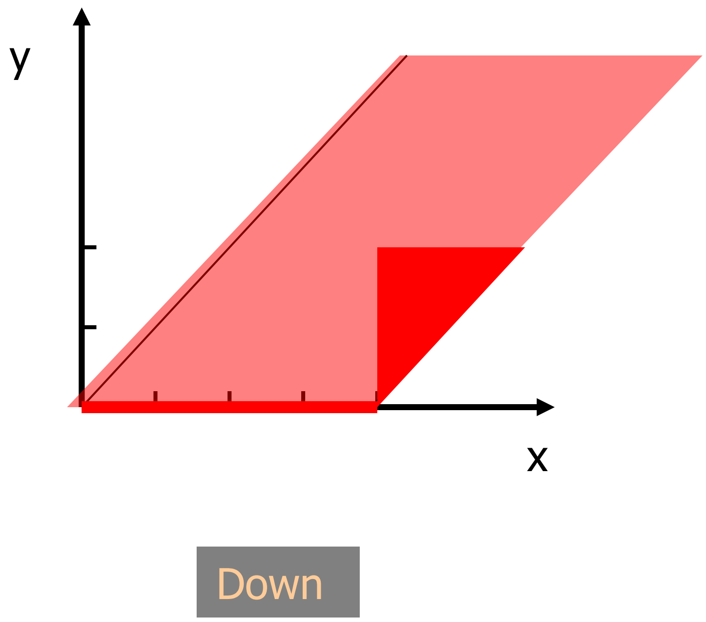
Symbolic Exploration





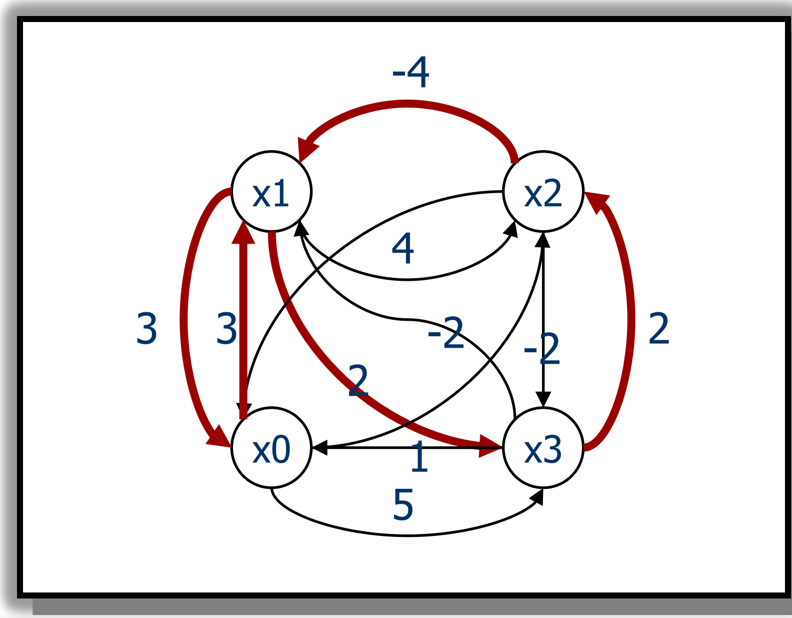
Symbolic Exploration

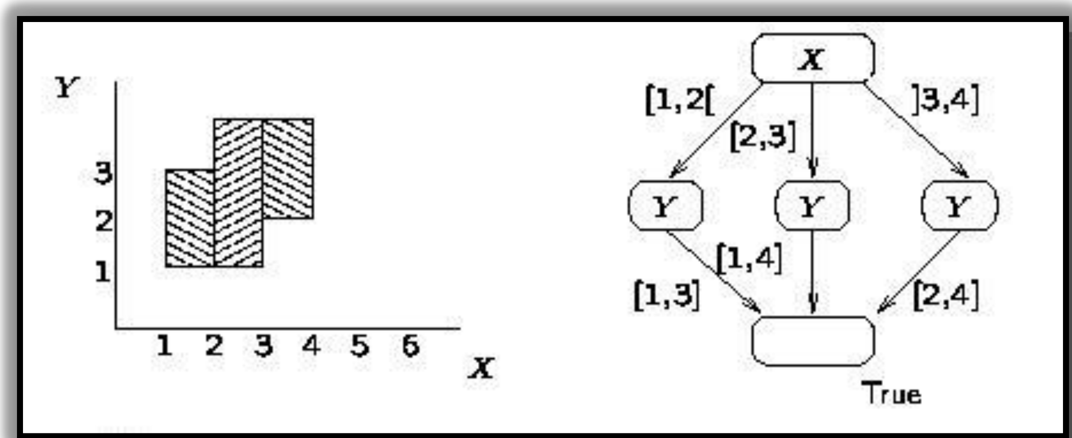




Datastructures for Zones

- Difference Bounded Matrices (DBMs)
- Minimal Constraint Form
  [RTSS97]
- Clock Difference Diagrams
  [CAV99]





Inclusion Checking (DBMs)

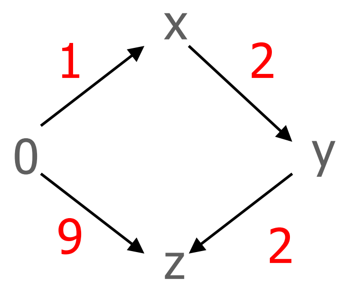
Bellman 1958, Dill 1989

Inclusion

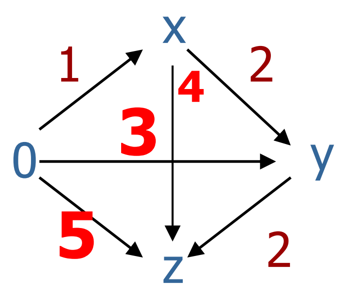
D1

x<=1
y-x<=2
z-y<=2
z<=9

Graph



Shortest Path Closure

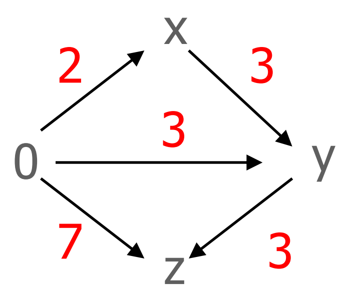


? ⊆ **?**

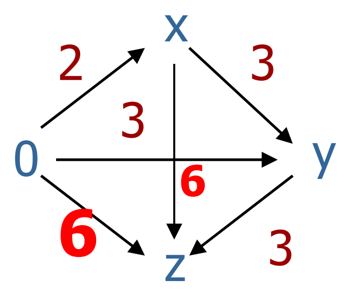
D2

x<=2
y-x<=3
y<=3
z-y<=3
z<=7

Graph

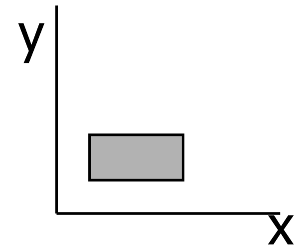


Shortest Path Closure

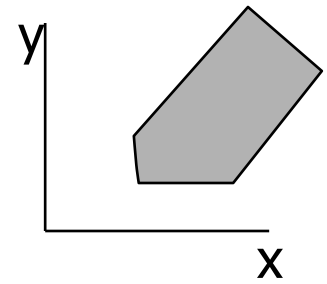


Future (DBMs)

D

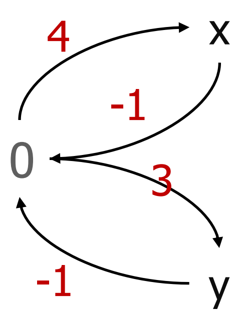


1<= x <=4
1<= y <=3

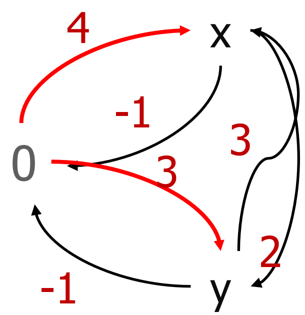


Future D

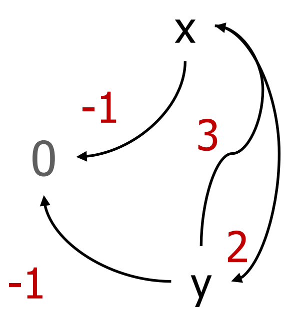
1<=x, 1<=y
-2<=x-y<=3



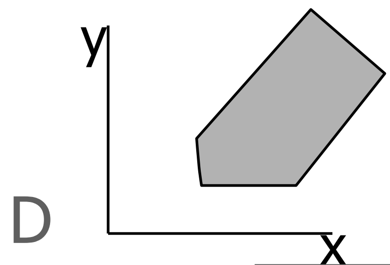
Shortest Path Closure



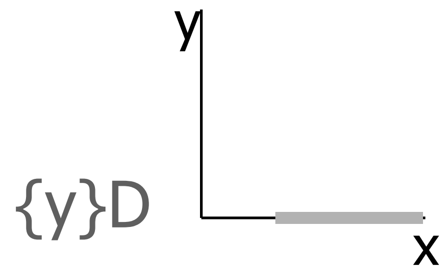
Remove upper bounds on clocks



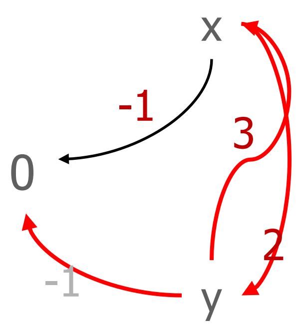
Reset (DBMs)



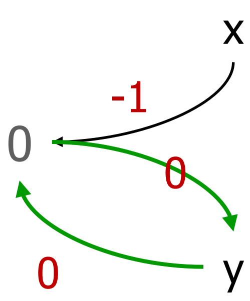
1<=x, 1<=y
-2<=x-y<=3



y=0, 1<=x

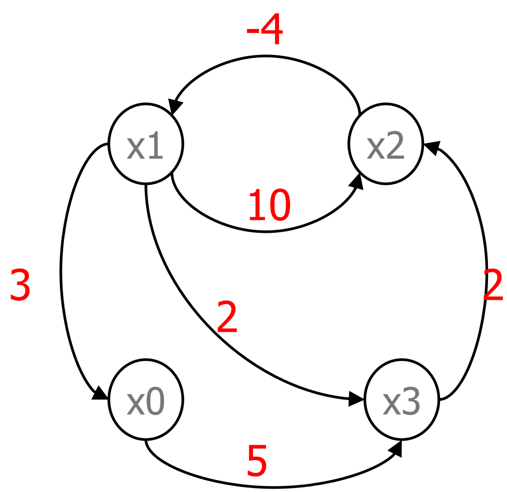


Remove all bounds involving y and set y to 0

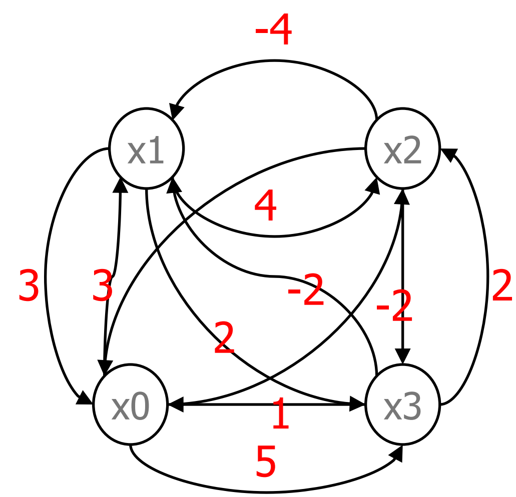


Difference Bounded Matrices

x1-x2<=4
x2-x1<=10
x3-x1<=2
x2-x3<=2
x0-x1<=3
x3-x0<=5



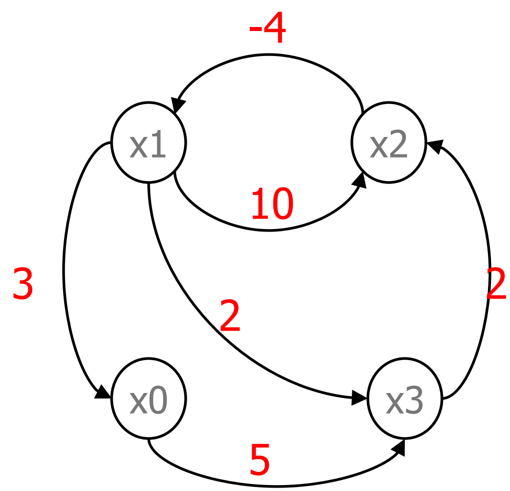
Shortest Path Closure O(n^3)



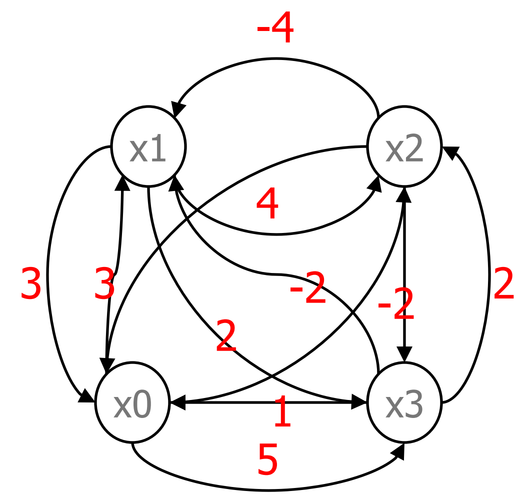
Minimal Constraint Form

RTSS 1997

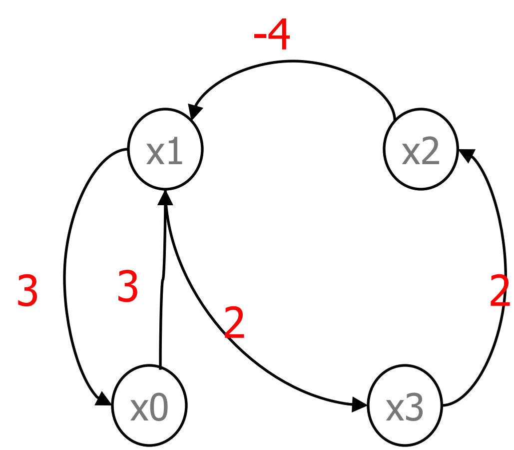
x1-x2<=4
x2-x1<=10
x3-x1<=2
x2-x3<=2
x0-x1<=3
x3-x0<=5



Shortest Path Closure O(n^3)



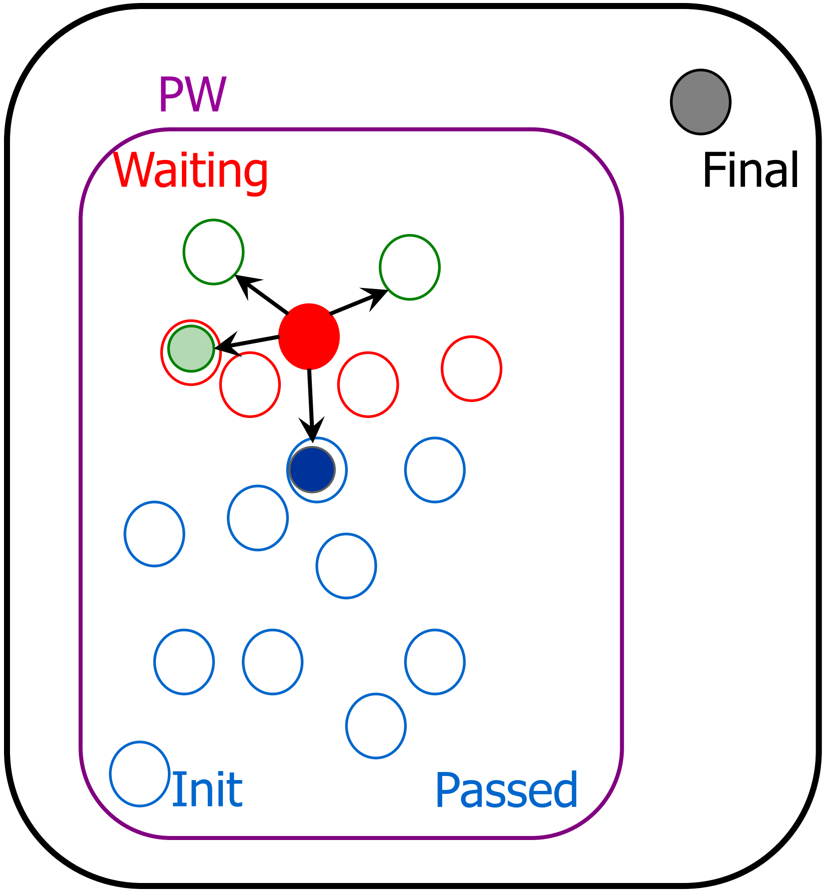
Shortest Path Reduction O(n^3)



Space worst O(n^2)
practice O(n)

Earlier Termination

Init -> Final ?



INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

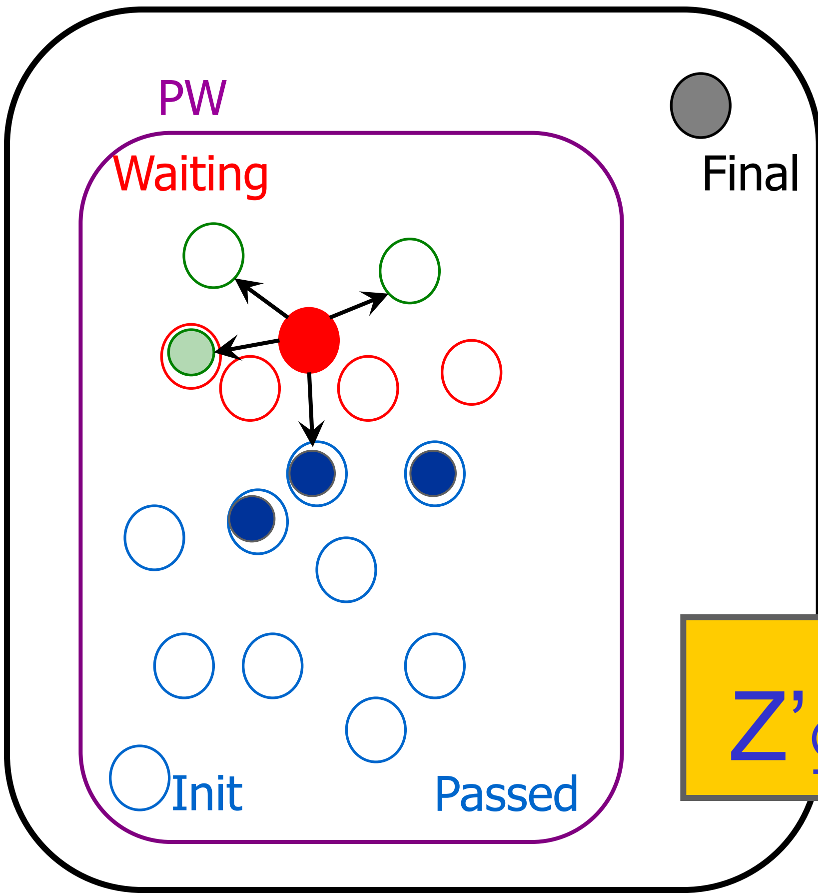
REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z)→(n',Z'):
if for some (n',Z'' ... tinue
else add (n',Z') to Waiting
move (n,Z) to Passed

$Z' \subseteq Z''$

UNTIL Waiting = Ø
return false

Earlier Termination

Init -> Final ?



INITIAL Passed := Ø;
Waiting := {(n_0,Z_0)}

REPEAT
pick (n,Z) in Waiting
if (n,Z) = Final return true
for all (n,Z)→(n',Z'):
if for some (n',Z' … tinue
else add (n',Z') to Waiting
move (n,Z) … ed
… ng = Ø

$Z' \subseteq Z''$

$Z' \subseteq \bigcup Z_i$

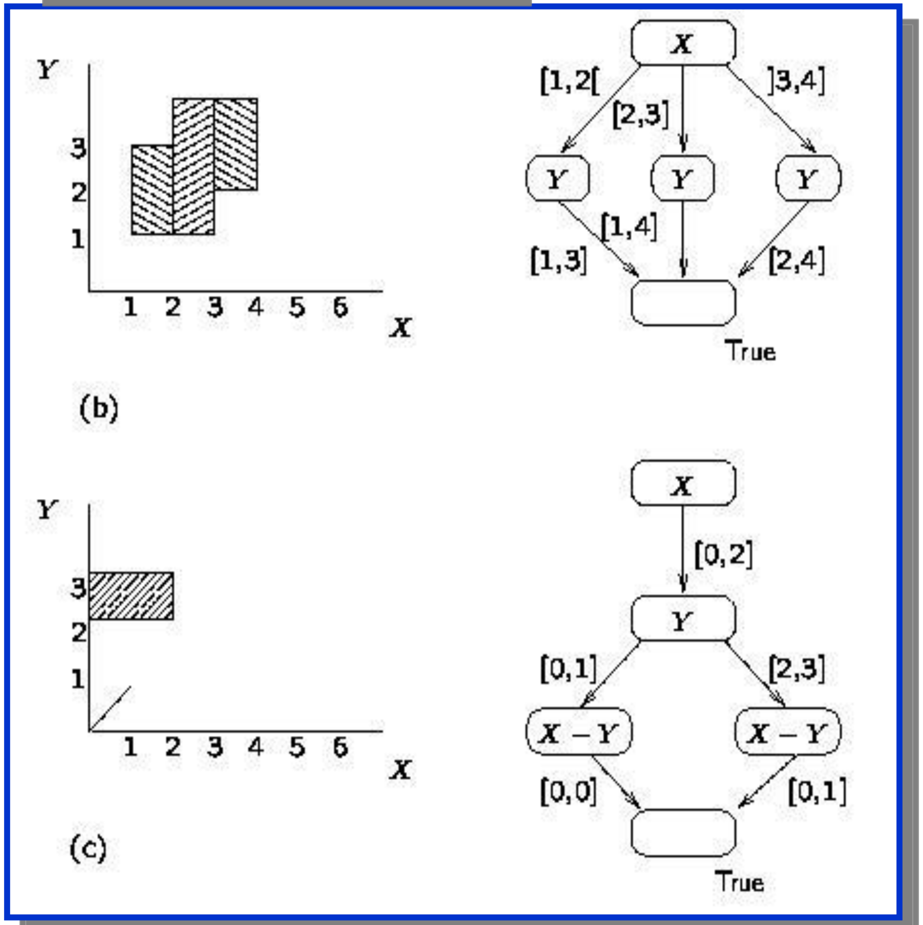
Clock Difference Diagrams

CAV99

- Nodes labeled with differences
- Maximal sharing of substructures (also across different CDDs)
- Maximal intervals
- Linear-time algorithms for set-theoretic operations.

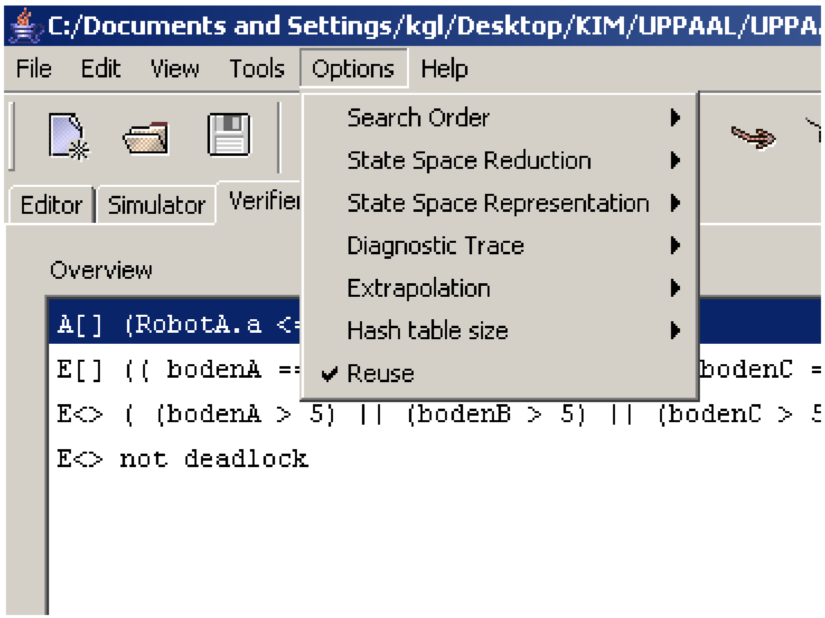
- NDD’s Maler et. al
- DDD’s Møller, Lichtenberg

CDD-representations



Verification Options

Verification Options



Search Order
- Depth First
- Breadth First

State Space Reduction
- None
- Conservative
- Aggressive

State Space Representation
- DBM
- Compact Form
- Under Approximation
- Over Approximation

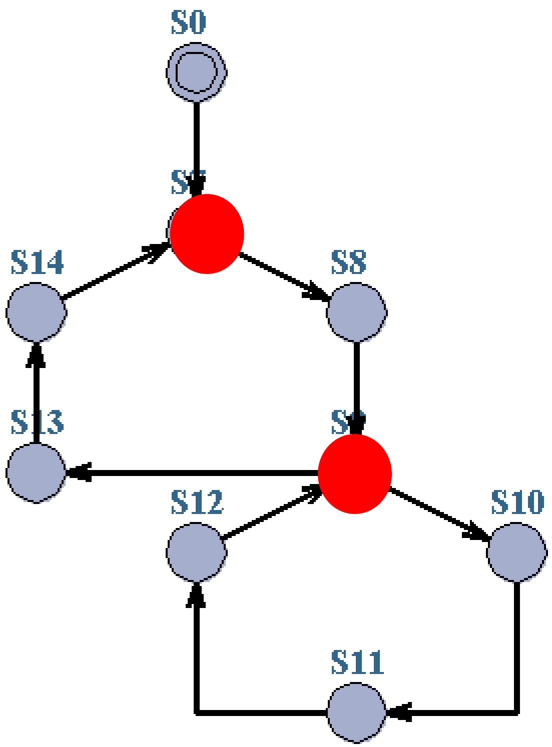
Diagnostic Trace
- Some
- Shortest
- Fastest

Extrapolation

Hash Table size

Reuse

State Space Reduction



Cycles:
Only symbolic states involving loop-entry points need to be saved on Passed list

To Store or Not To Store

Behrmann, Larsen, Pelanek 2003

117 states$_{\text{total}}$

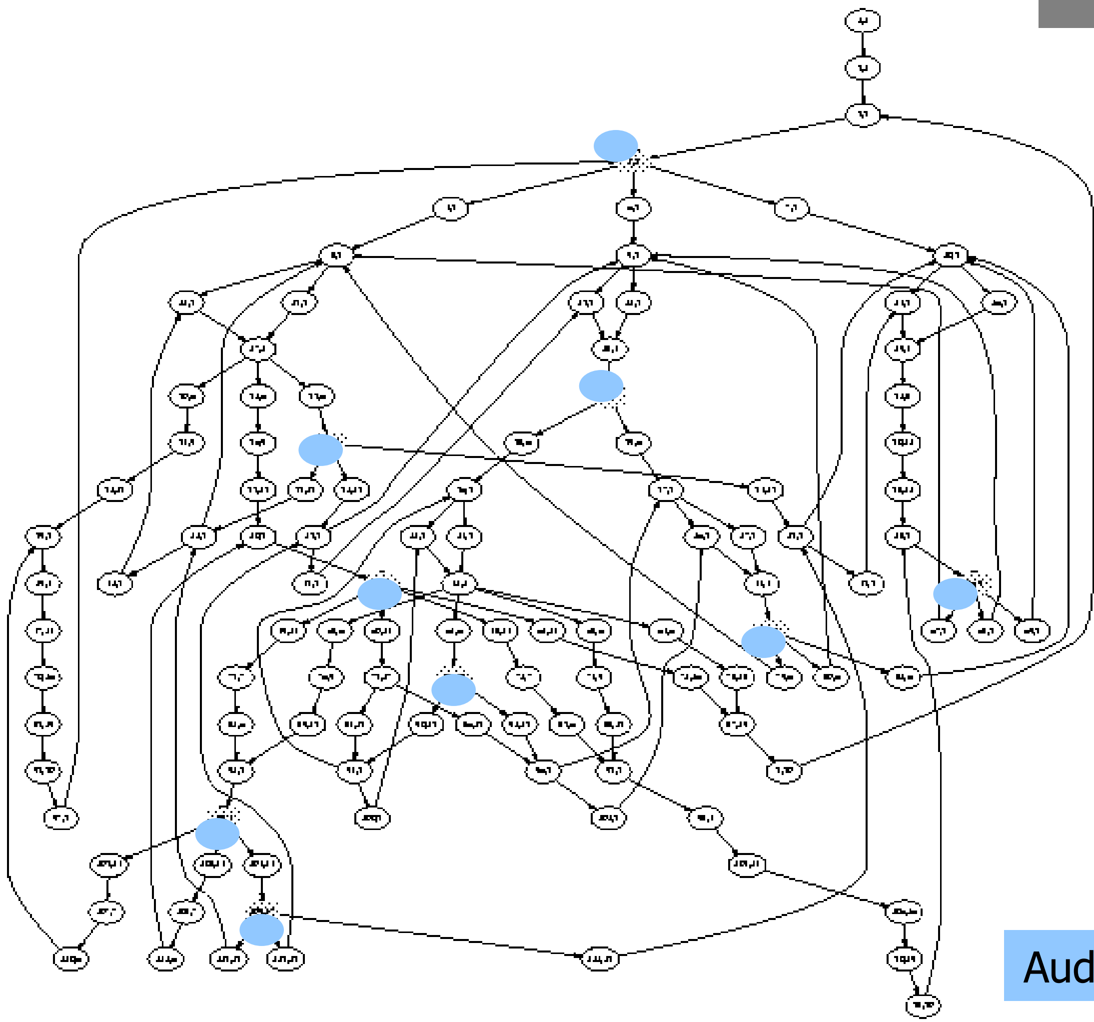
$\rightarrow$

81 states$_{\text{entrypoint}}$

$\rightarrow$

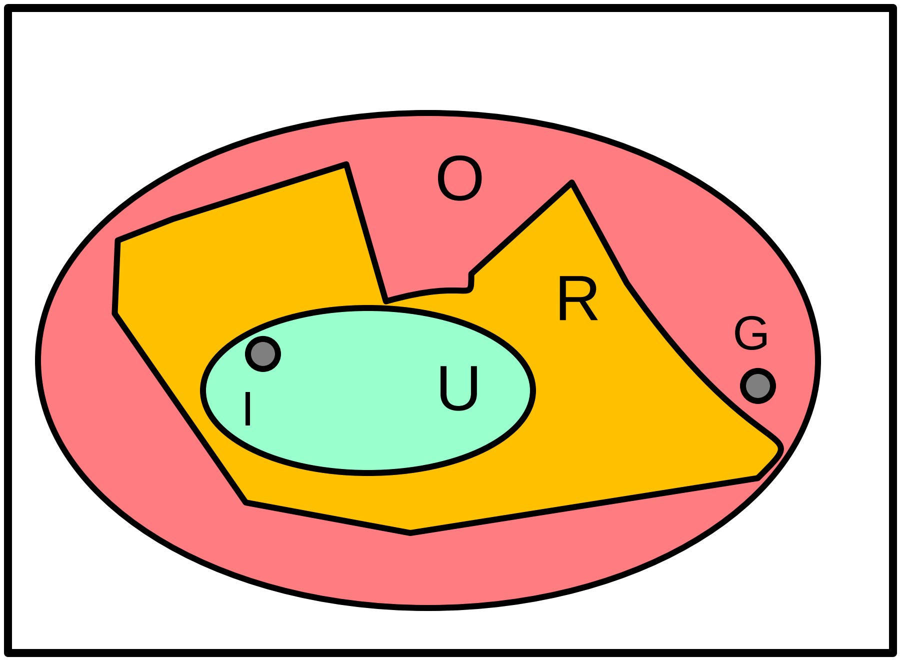
9 states

Time OH
less than 10%



Audio Protocol

Over/Under Approximation



Declared State Space

Question:

$G \in R$?

How to use:

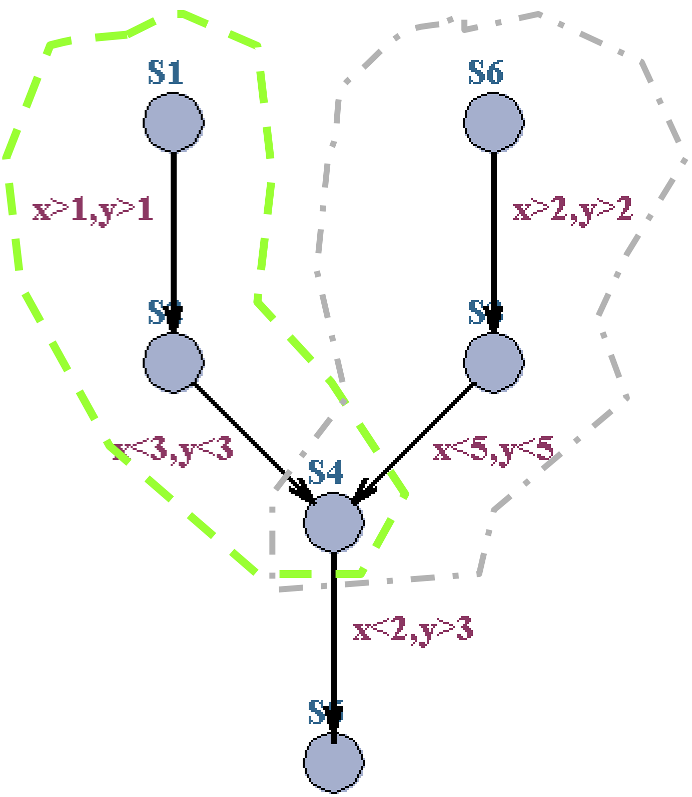
$G \in O$?

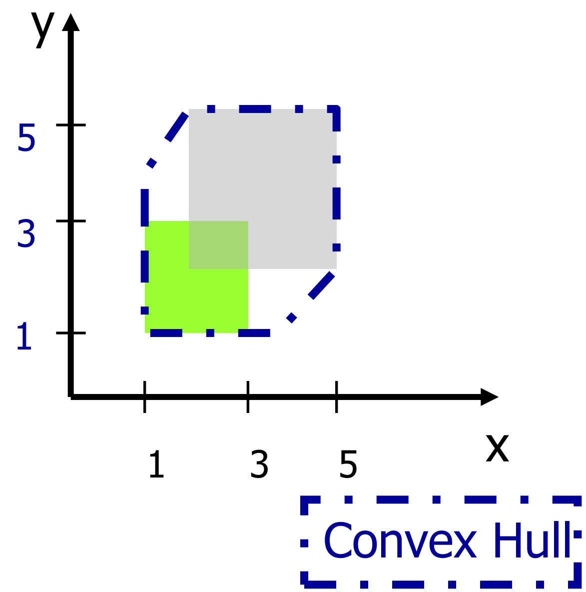
$G \in U$?

$G \in U \Rightarrow G \in R$

$\neg(G \in O) \Rightarrow \neg(G \in R)$

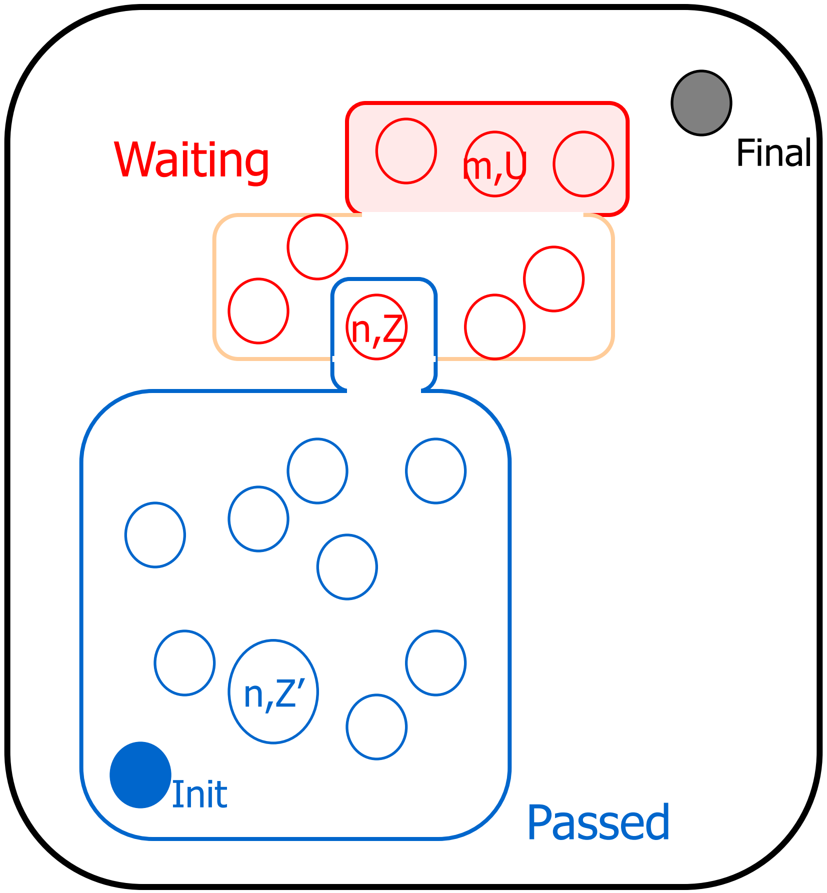
Over-approximation Convex Hull



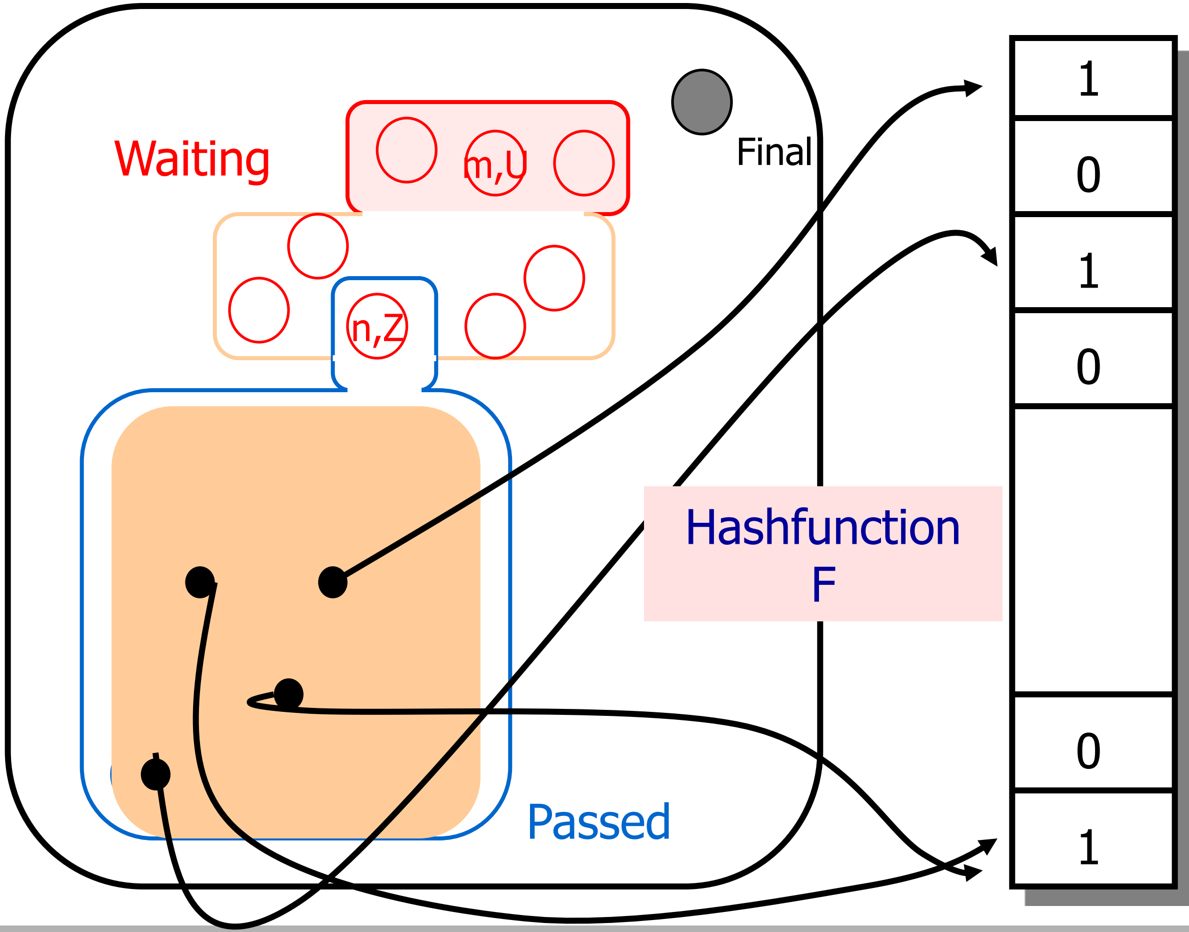


TACAS04: An EXACT method performing as well as Convex Hull has been developed based on abstractions taking max constants into account distinguishing between clocks, locations and $\leq$ & $\geq$

Under-approximation
Bitstate Hashing



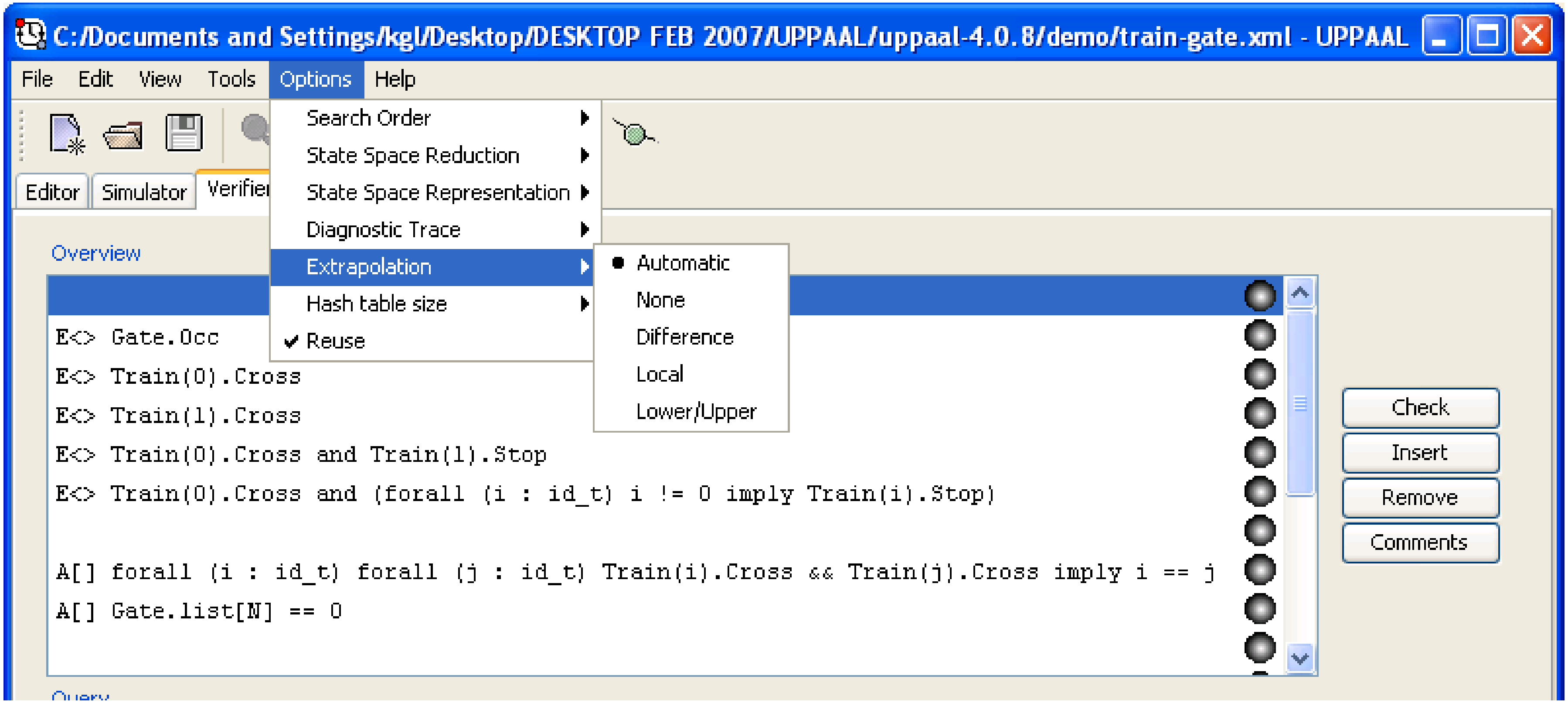
Under-approximation Bitstate Hashing



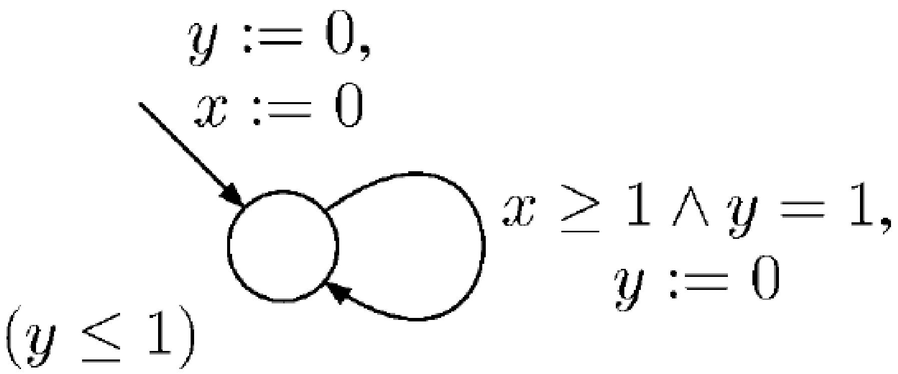
Passed= Bitarray

UPPAAL
4 - 512 Mbits

Extrapolation

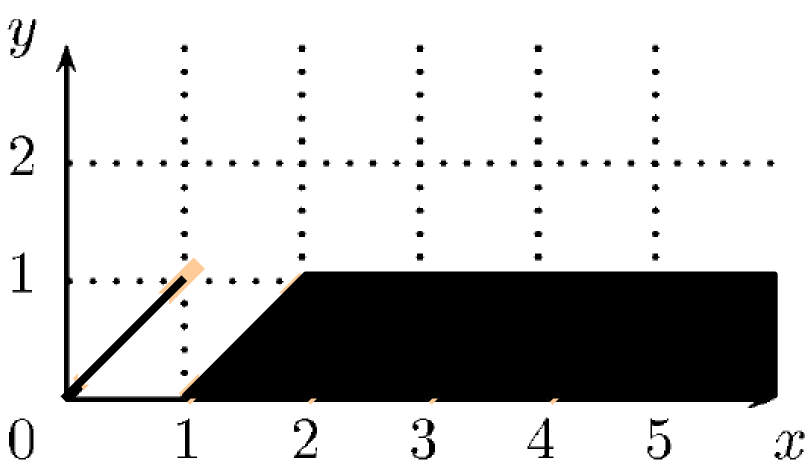


Forward Symbolic Exploration



TERMINATION not garanteed

Need for Finite Abstractions



Abstractions

$$a : \mathcal{P}(R_{\geq 0}^{X}) \hookrightarrow \mathcal{P}(R_{\geq 0}^{X}) \text{ such that } W \subseteq a(W)$$

$$\frac{(\ell, W) \Rightarrow (\ell', W')}{(\ell, W) \Rightarrow_a (\ell', a(W'))} \quad \text{if } W = a(W)$$

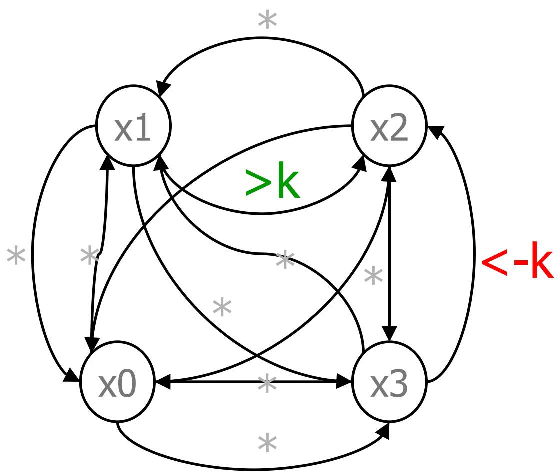
We want $\Rightarrow_a$ to be:

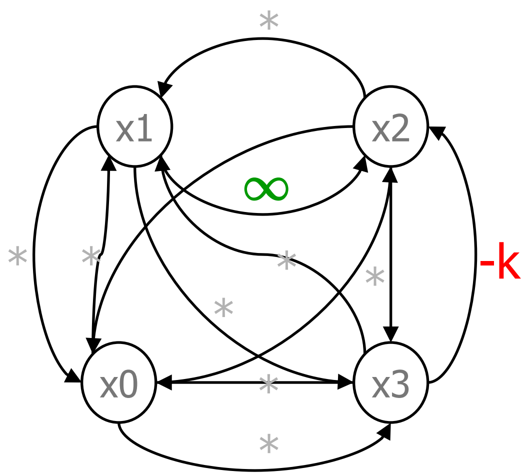
- sound & complete wrt reachability
- finite
- easy to compute
- as coarse as possible

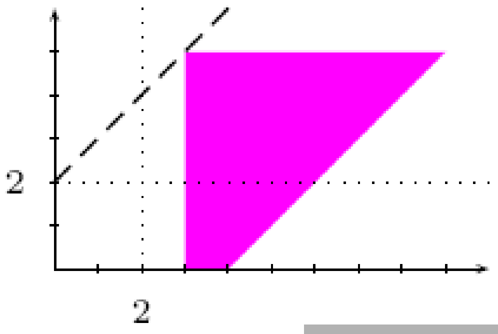
Abstraction by Extrapolation

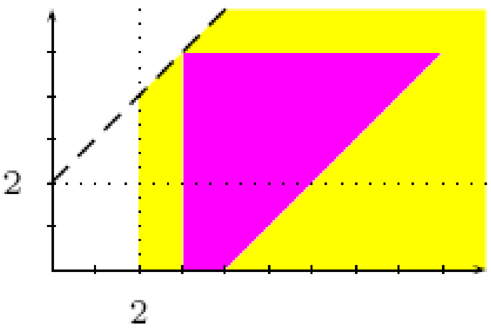
[Daws,Tripakis 98]

Let *k* be the largest constant appearing in the TA







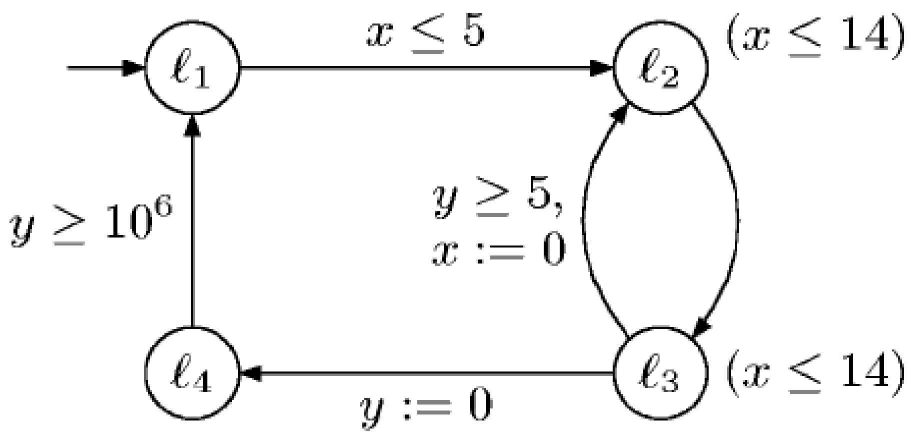


Sound & Complete

Ensures Termination

Location Dependency

[Behrmann, Bouyer, Fleury, Larsen 03]



$k_x = 5 \quad k_y = 10^6$

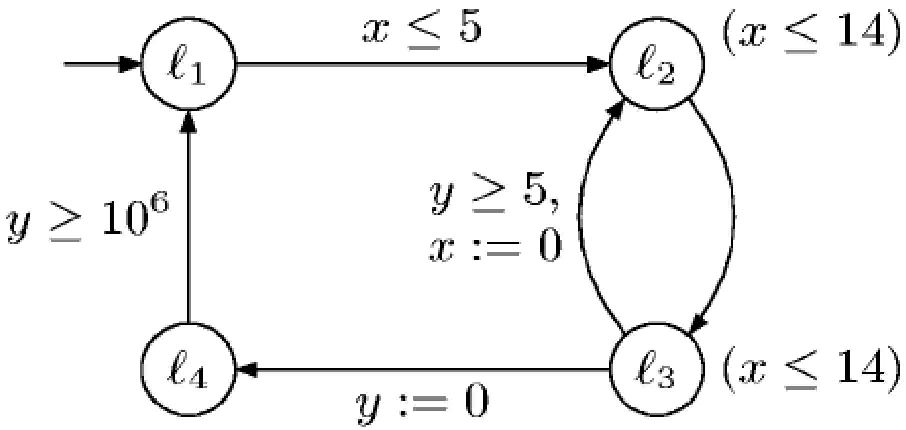
Will generate all symbolic states of the form

$(l_2, x \in [0,14], y \in [5,14n], y-x \in [5,14n-14])$

for $n \leq 10^6/14$!!

But $y \geq 10^6$ is not RELEVANT in l_2

Location Dependent Constants



$k_x = 5 \quad k_y = 10^6$

k_x^i	$= 14$	for $i \in \{1,2,3,4\}$
k_y^i	$= 5$	for $i \in \{1,2,3\}$
k_y^4	$= 10^6$	

k_j^i may be found as solution to simple linear constraints!

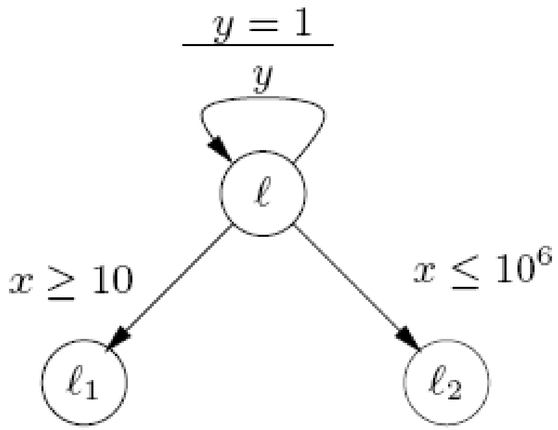
Active Clock Reduction:
$k_j^i = -\infty$

Experiments

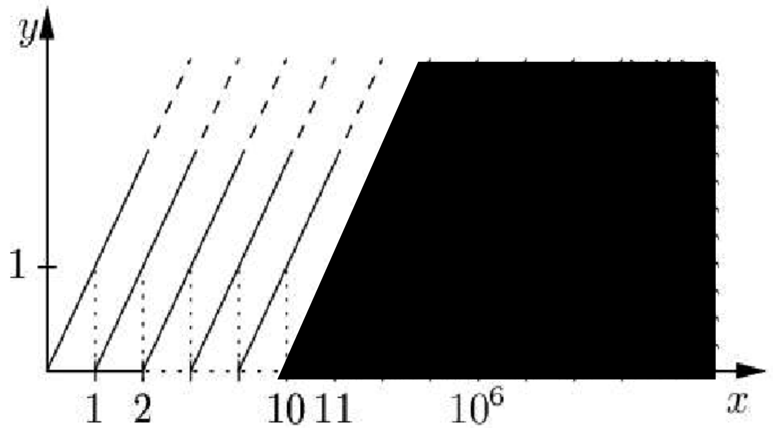
Active by default ↓

	Constant BIG	Global Method	Active-clock Reduction	Local Constants
Naive Example	10^3	0.05s/1MB	0.05s/1MB	0.00s/1MB
	10^4	4.78s/3MB	4.83s/3MB	0.00s/1MB
	10^5	484s/13MB	480s/13MB	0.00s/1MB
	10^6	stopped	stopped	0.00s/1MB
Two Processes	10^3	3.24s/3MB	3.26s/3MB	0.01s/1MB
	10^4	5981s/9MB	5978s/9MB	0.37s/2MB
	10^5	stopped	stopped	72s/5MB
Asymmetric Fischer	10^3	0.01s/1MB	0.01s/1MB	0.01s/1MB
	10^4	2.20s/3MB	2.20s/3MB	0.85s/2MB
	10^5	333s/19MB	333s/19MB	160s/13MB
	10^6	33307s/122MB	33238s/122MB	16330s/65MB
Bang & Olufsen	25000	stopped	159s/243MB	123s/204MB

Lower and Upper Bounds [Behrmann, Bouyer, Larsen, Pelanek 04]



$k_x^{l} = 10^6$



Given that $x \leq 10^6$ is an *upper* bound implies that

(l, v_x, v_y) *simulates* (l, v'_x, v_y)

whenever $v'_x \geq v_x \geq 10$.

For reachability downward closure wrt simulation suffices!

Advanced Extrapolation

		Classical			Loc. dep. Max			Loc. dep. LU			Convex Hull		
		-n1			-n2			-n3			-A		
	Model	Time	States	Mem	Time	States	Mem	Time	States	Mem	Time	States	Mem
Fischer	f5	4.02	82,685	5	0.24	16,980	3	0.03	2,870	3	0.03	3,650	3
	f6	597.04	1,489,230	49	6.67	158,220	7	0.11	11,484	3	0.10	14,658	3
	f7				352.67	1,620,542	46	0.47	44,142	3	0.45	56,252	5
	f8							2.11	164,528	6	2.08	208,744	12
	f9							8.76	598,662	19	9.11	754,974	39
	f10							37.26	2,136,980	68	39.13	2,676,150	143
	f11							152.44	7,510,382	268			
CSMA/CD	c5	0.55	27,174	3	0.14	10,569	3	0.02	2,027	3	0.03	1,651	3
	c6	19.39	287,109	11	3.63	87,977	5	0.10	6,296	3	0.06	4,986	3
	c7				195.35	813,924	29	0.28	18,205	3	0.22	14,101	4
	c8							0.98	50,058	5	0.66	38,060	7
	c9							2.90	132,623	12	1.89	99,215	17
	c10							8.42	341,452	29	5.48	251,758	49
	c11							24.13	859,265	76	15.66	625,225	138
	c12							68.20	2,122,286	202	43.10	1,525,536	394
	bus	102.28	6,727,443	303	66.54	4,620,666	254	62.01	4,317,920	246	45.08	3,826,742	324
	philips	0.16	12,823	3	0.09	6,763	3	0.09	6,599	3	0.07	5,992	3
	sched	17.01	929,726	76	15.09	700,917	58	12.85	619,351	52	55.41	3,636,576	427

Additional “secrets”

- Sharing among symbolic states
  - location vector / discrete values / zones
- Symmetry Reduction
- Sweep Line Method
- Guiding wrt Heuristic Value (CORA)
  - User-supplied / Auto-generated
- “Manual” tricks:
  - active variable reduction
  - Value passing using arrays of channels

Open Problems

- Fully symbolic exploration of TA (both discrete and continuous part) ?
- Canonical form for CDD's ?
- Partial Order Reduction ?
- Compositional Backwards Reachability ?
- Bounded Model Checking for TA ?
- Exploitation of multi-core processors ?
- …

Application: Schedulability Analysis

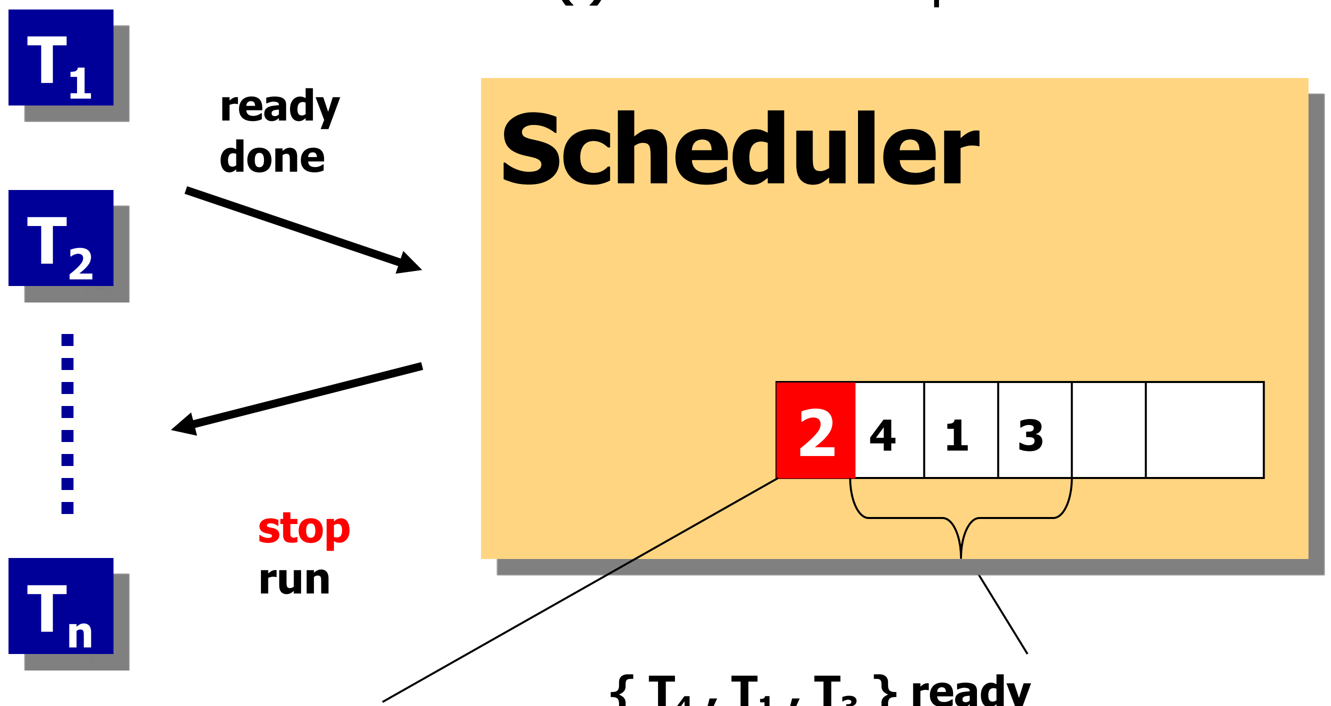
Task Scheduling

utilization of CPU

P(i), [E(i), L(i)], .. : period or earliest/latest arrival or .. for T_i

C(i): execution time for T_i

D(i): deadline for T_i



T_2 is running

{ T_4 , T_1 , T_3 } ready ordered according to some given priority: (e.g. Fixed Priority, Earliest Deadline,..)

Classical Scheduling Theory

Utilisation-Based Analysis

- A simple sufficient but not necessary schedulability test exists

$$U \equiv \sum_{i=1}^{N} \frac{C_i}{T_i} \leq N(2^{1/N} - 1)$$

$$U \leq 0.69 \text{ as } N \rightarrow \infty$$

Where C is WCET and T is period

41

Response Time Equation

$$R_i = C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Where $hp(i)$ is the set of tasks with priority higher than task i

Solve by forming a recurrence relationship:

$$w_i^{n+1} = C_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{T_j} \right\rceil C_j$$

The set of values $w_i^0, w_i^1, w_i^2, \ldots, w_i^n, \ldots$ is monotonically non decreasing
When $w_i^n = w_i^{n+1}$ the solution to the equation has been found, w_i^0 must not be greater that R_i (e.g. 0 or C_i)

42

Classical WCRT Analysis

Quasimodo

- "Classical" scheduling analysis technique
- *For all tasks i: $WCRT_i \leq Deadline_i$*

$$R_i = B_i + C_i + \sum_{j \in hp(i)} \left\lceil \frac{R_i}{T_j} \right\rceil C_j$$

Blocking times for priority inheritance protocol (BSW):

$$Blocking(i) = \sum_{r=1}^{R} usage(r,i) WCET_{CriticalSection}(r)$$

Blocking times for priority ceiling protocol (ASW):

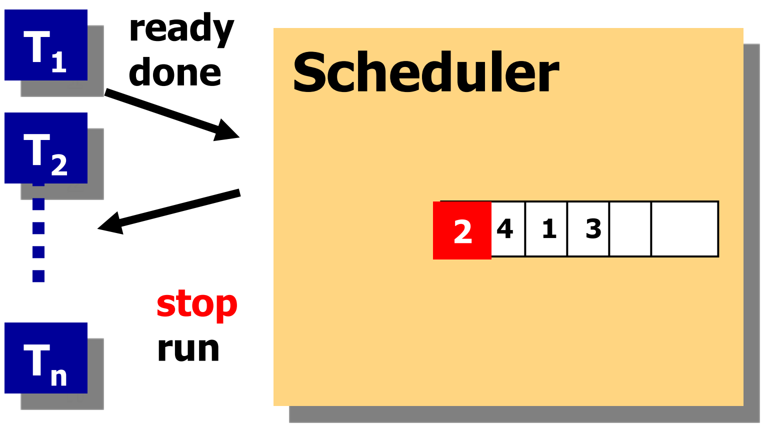
$$Blocking(i) = \max_{r=1}^{R} usage(r,i) WCET_{CriticalSection}(r)$$

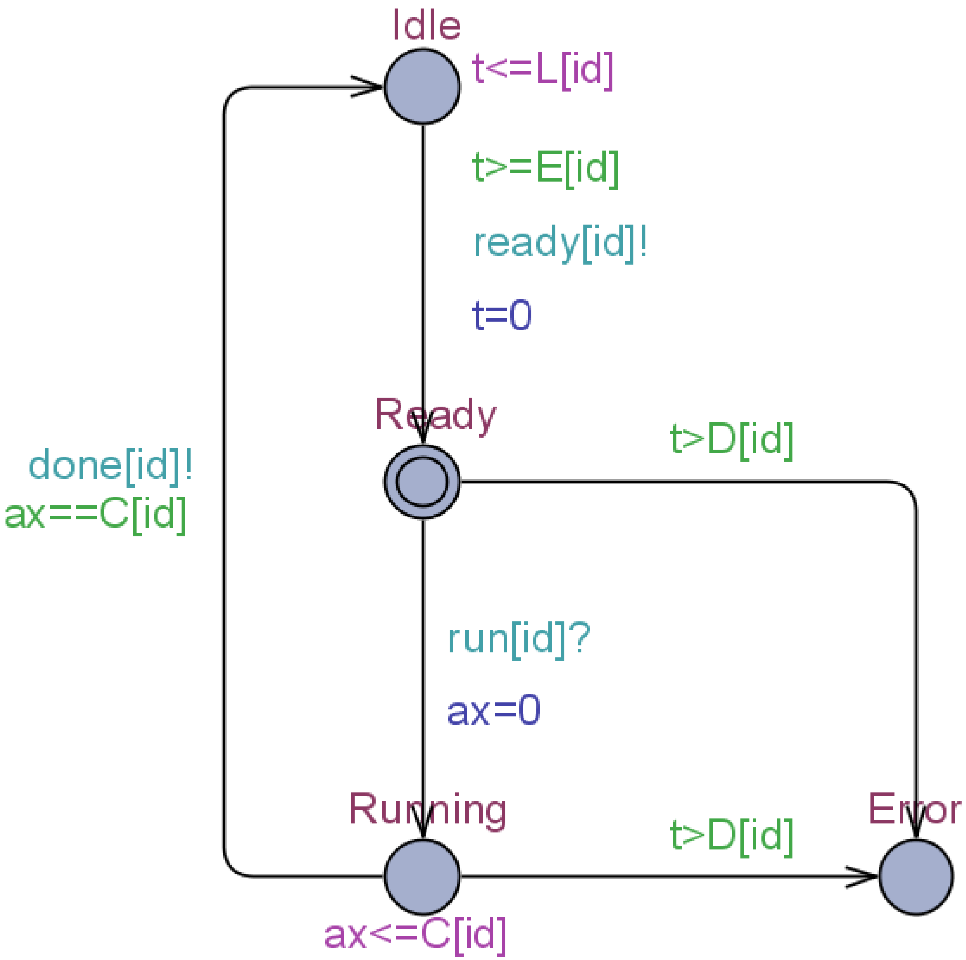
Quasimodo Workshop, Eindhoven, Nov 6, 2009 Page 21

✓ Simple to perform

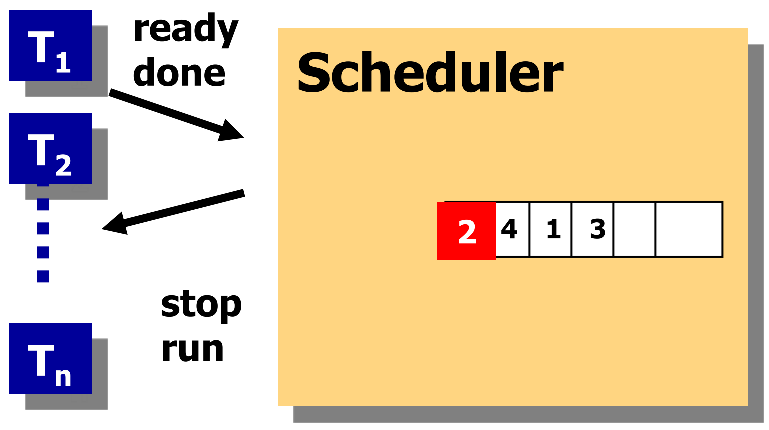
– Overly conservative
– Limited settings
– Single-processor
⇒ Do it in UPPAAL!

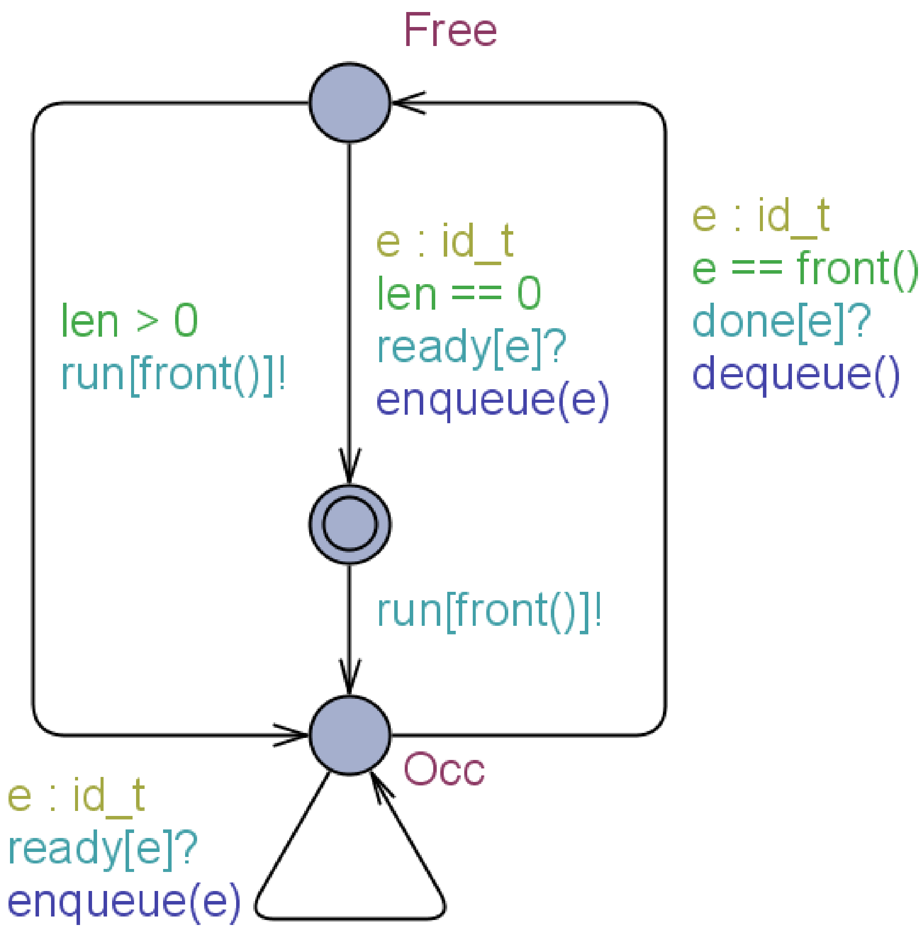
Modeling Task





Modeling Scheduler

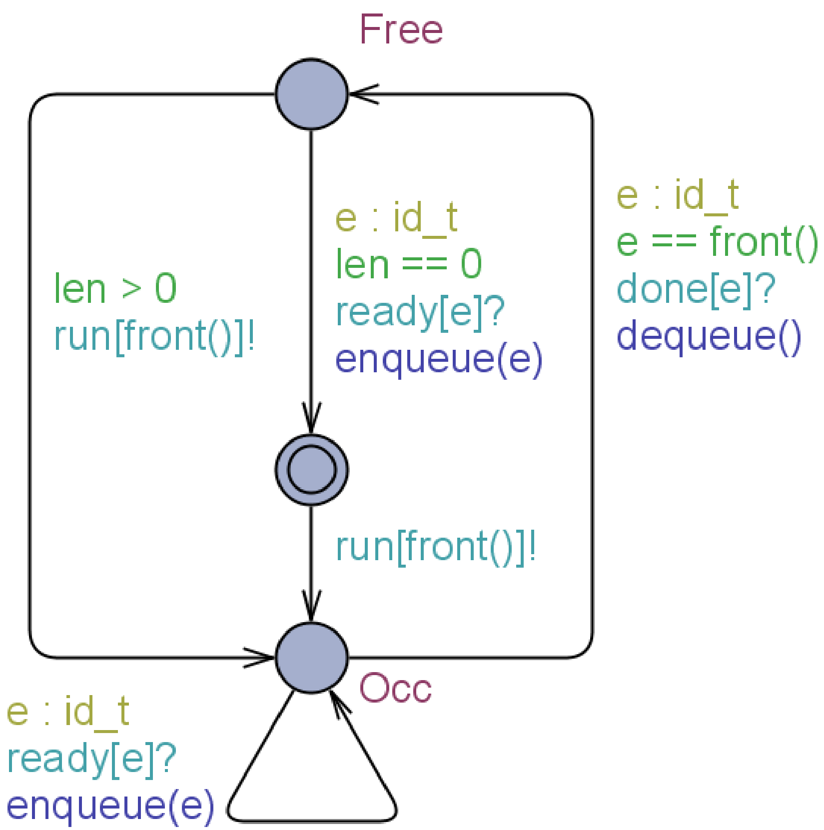




Implementation of enqueue/dequeue
⇒ scheduling policy

Modeling Queue

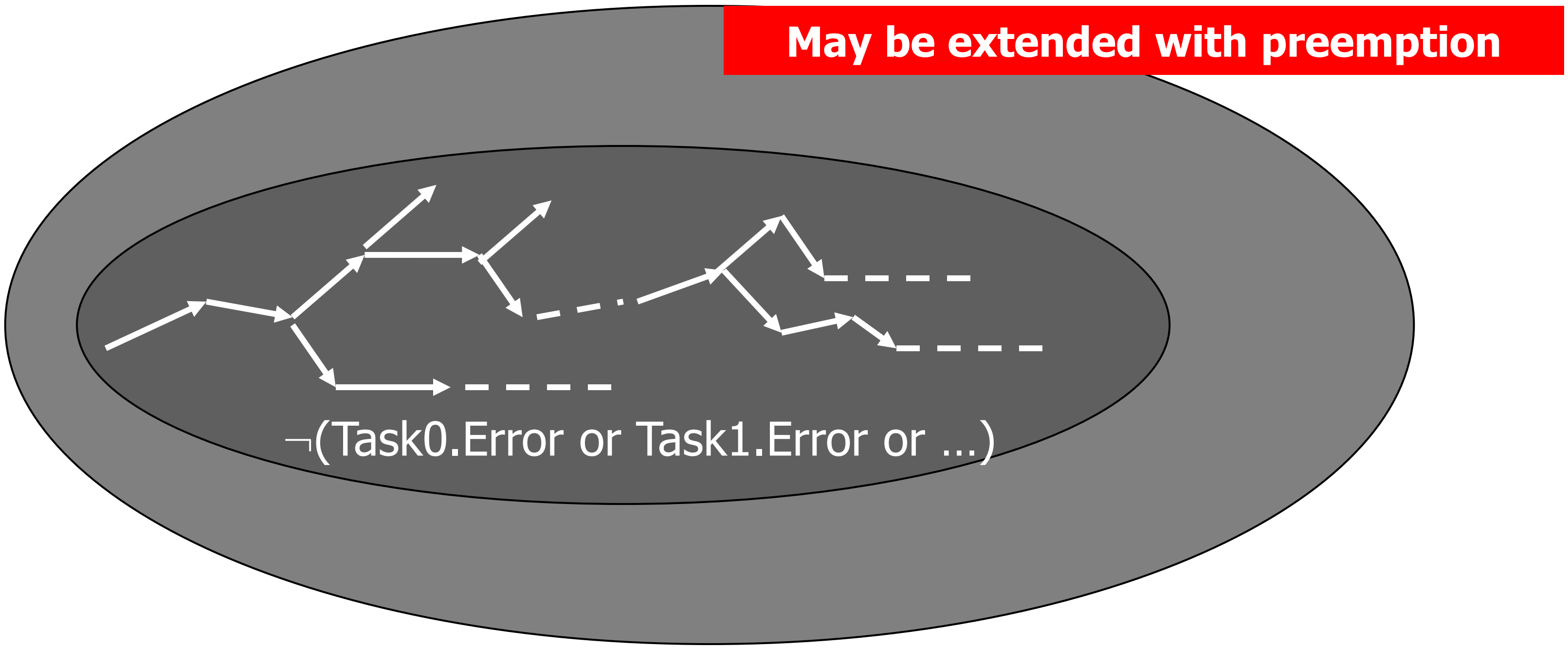
In UPPAAL 4.0
User Defined Function



Sort by priority

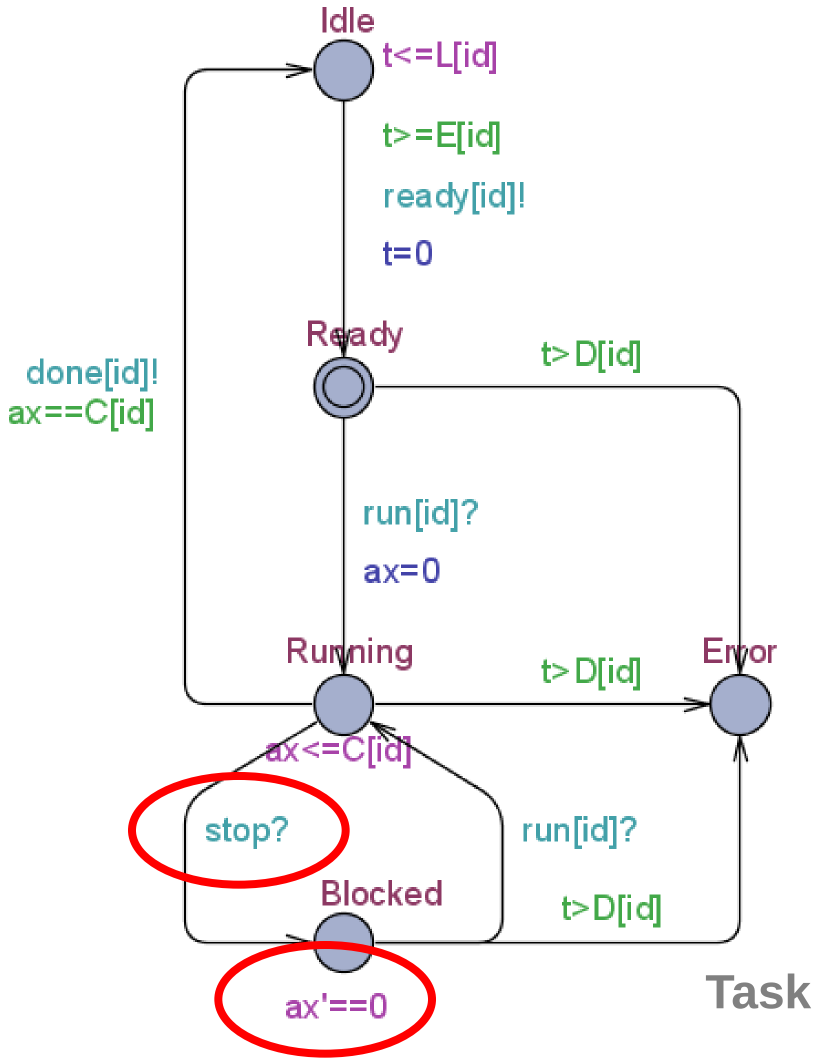
```
// Put an element at the end of the queue
void enqueue(id_t element)
{
int tmp=0;
list[len++] = element;
if (len>0)
{
        int i=len-1;
        while (i>1 && P[list[i]]>P[list[i-1]])
        {
                tmp = list[i-1];
                list[i-1] = list[i];
                list[i] = tmp;
                i--;
        }
}
}

// Remove the front element of the queue
void dequeue()
{ ......
```

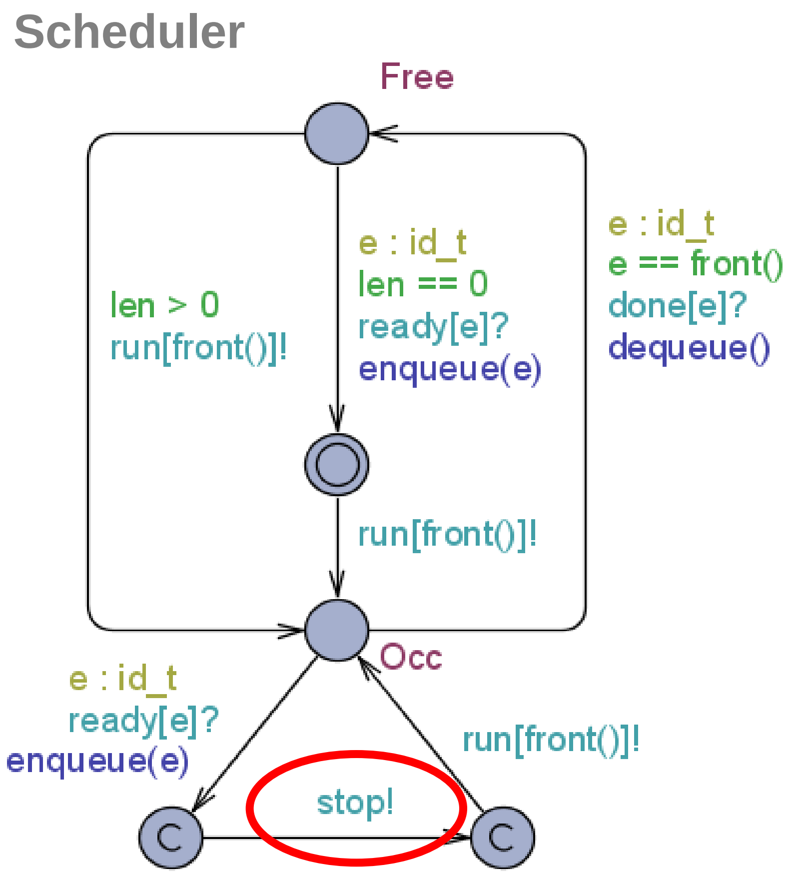
Schedulability = Safety Property



A□ ¬(Task0.Error or Task1.Error or …)

Preemption – Stopwatches!





Defeating undecidability ☺

Stop-Watches

- Make reachability undecidable.
- Over-approximation used in UPPAAL
  - $\Rightarrow$ Safe for positive schedulability results!

- What to do if you violate deadlines?
  - Try to validate the trace using other techniques, e.g., polyhedra.
  - Use SMC!

LAB-Exercises (cont)

www.cs.aau.dk/~kgl/Shanghai2013/exercises

Exercise 1 (Brick Sorter)

Exercise 2 (Coffee Machine)

Excercise 19 (Train Crossing)

Exercise 28 (Jobshop Scheduling)

Exercise 14 (Gossiping Girls)